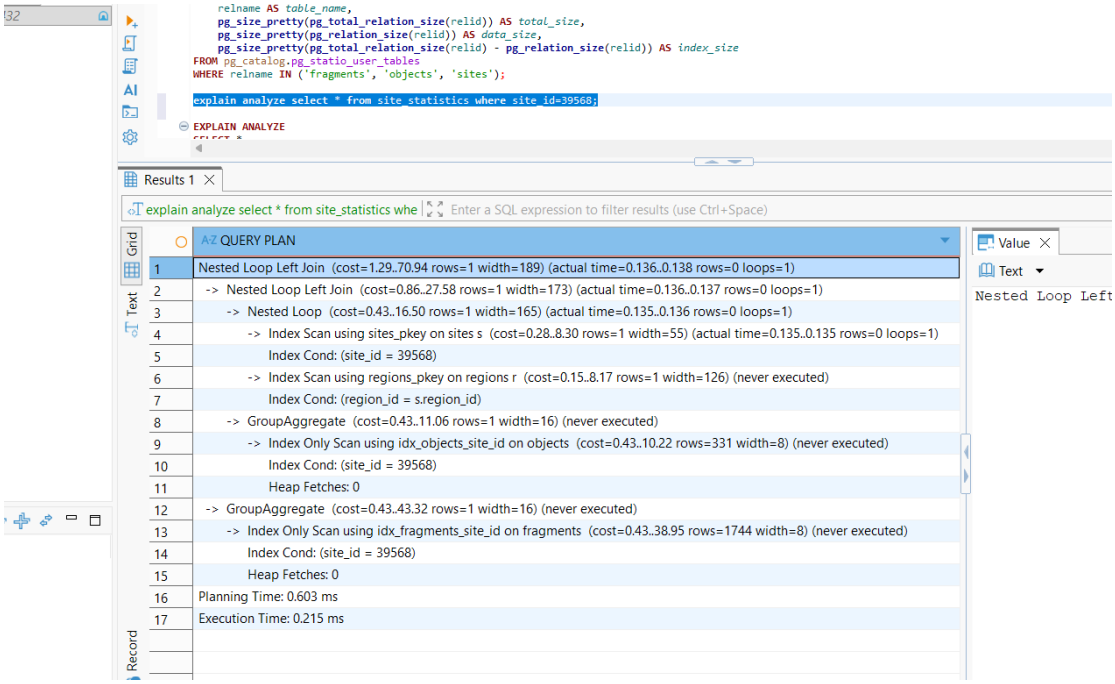
# Напредни бази на податоци
# Фаза 3 - Индекси и оптимизација на прашалници

## Проект: HeritageGuard

Паула Коцева 231130
Елеонора Маркоска 231256
Ивана Павлова 231511

### View 1 - Site_Statistics




Овој поглед е наменет за административен и аналитички преглед на статистички информации за секој археолошки локалитет во базата. Неговата главна цел е да овозможи брз и едноставен пристап до клучни податоци за конкретен локалитет преку пребарување според site_id, без потреба од пишување комплексни SQL барања над повеќе табели.

Преку овој view, корисникот може да добие интегриран приказ кој ги содржи основните информации за локалитетот (site_id, site_name, region), како и агрегирани статистики за:

- вкупен број на археолошки објекти (total_objects)
- вкупен број на фрагменти (total_fragments)

На овој начин, погледот служи како централизирана точка за анализа на состојбата на локалитетите, што е особено корисно за истражувачи, администратори и институции кои вршат следење на археолошки ресурси.

Во позадина, view-от користи LEFT JOIN со агрегирани подзапити (COUNT ... GROUP BY) над табелите Objects и Fragments, со што се овозможува приказ и на локалитети кои немаат регистрирани објекти или фрагменти. Функцијата COALESCE се користи за замена на NULL вредности со 0, со што резултатите стануваат поконзистентни и попогодни за анализа.

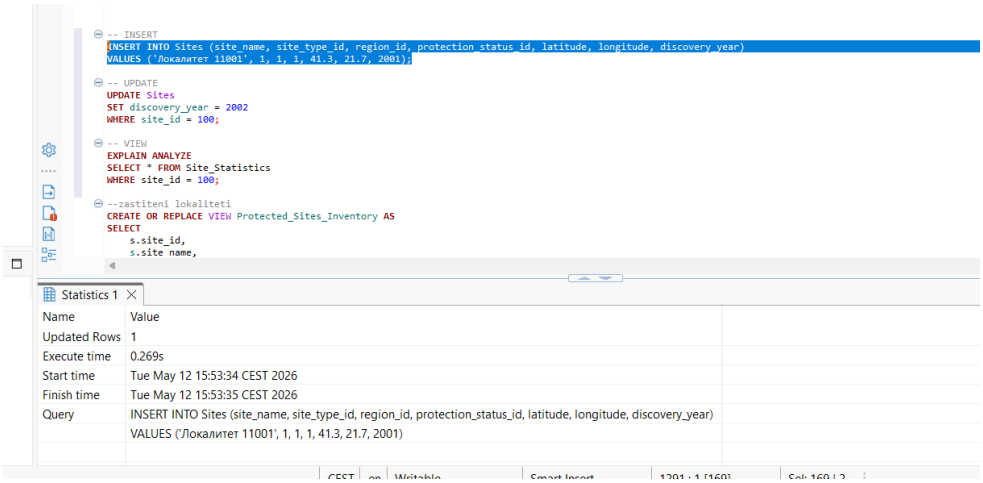
За подобрување на перформансите при пребарување и агрегација по site_id, имплементирани се следните индекси:

```
CREATE INDEX idx_objects_site_id ON Objects(site_id);
CREATE INDEX idx_fragments_site_id ON Fragments(site_id);
```

Овие индекси значително го намалуваат времето потребно за извршување на пребарувања и статистички пресметки, особено поради големиот обем на податоци во табелите Objects и Fragments, бидејќи PostgreSQL може да користи Index Scan наместо Sequential Scan.

Како резултат, Site_Statistics претставува оптимизиран аналитички поглед кој ја поедноставува обработката на податоци, ја подобрува читливоста, овозможува повторна употреба на логиката и обезбедува ефикасен мониторинг на археолошките локалитети.

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

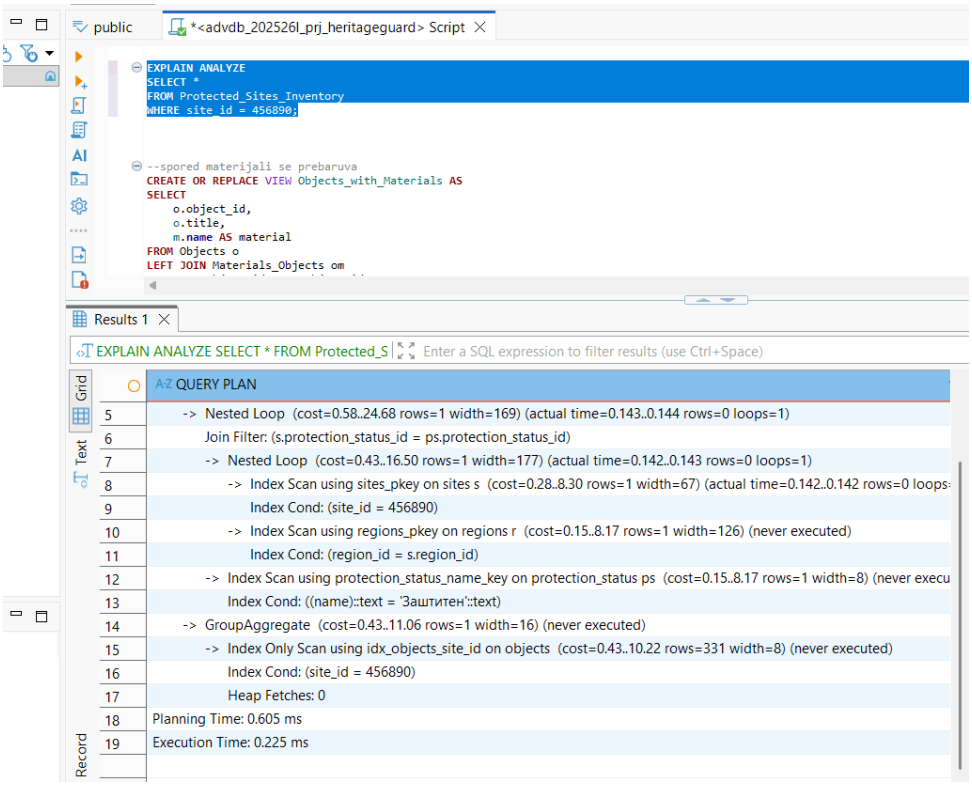


```
BEGIN;
SET LOCAL synchronous_commit = OFF;

-- INSERT
INSERT INTO Sites (site_name, site_type_id, region_id, protection_status_id, latitude, longitude, discovery_year)
VALUES ('Локалитет 11001', 1, 1, 1, 41.3, 21.7, 2001);

-- UPDATE
UPDATE Sites
SET discovery_year = 2002
WHERE site_id = 100;

-- UPDATE
UPDATE Sites
SET discovery_year = 2002
WHERE site_id = 100;

-- VIEW
EXPLAIN ANALYZE
SELECT * FROM Site_Statistics
WHERE site_id = 100;
```

Statistics 1

| Name | Value |
|---|---|
| Updated Rows | 1 |
| Execute time | 0.426s |
| Start time | Tue May 12 16:08:04 CEST 2026 |
| Finish time | Tue May 12 16:08:04 CEST 2026 |
| Query | UPDATE Sites |
| | SET discovery_year = 2002 |
| | WHERE site_id = 100 |

View 2 - Protected_Sites_Inventory




Овој поглед е наменет за административен, институционален и аналитички преглед на сите археолошки локалитети кои имаат статус „Заштитен“. Неговата основна цел е да овозможи брз пристап до информации за заштитените

локалитети и нивниот археолошки инвентар, без потреба од сложени SQL пребарувања низ повеќе поврзани табели.

Преку овој view, корисникот може да добие структуриран приказ за секој заштитен локалитет, кој ги содржи следните информации:

- идентификатор на локалитет (site_id)
- име на локалитет (site_name)
- регион (region)
- година на откривање (discovery_year)
- вкупен број на објекти поврзани со локалитетот (total_objects_count)

Овој поглед е особено корисен за државни институции, музеи, истражувачи и организации задолжени за заштита на културното наследство, бидејќи овозможува систематски увид во археолошките ресурси на локалитетите со највисок степен на заштита.

Во неговата имплементација се користи JOIN помеѓу табелите Sites, Regions и Protection_Status, при што клучниот филтер е:

```
WHERE ps.name = 'Заштитен'
```

Со ова се обезбедува приказ исклучиво на локалитети со активен статус на заштита. Дополнително, преку LEFT JOIN со агрегирана подтабела над Objects, се пресметува бројот на објекти по локалитет:

```
SELECT site_id, COUNT(*) AS total_objects_count
FROM Objects
GROUP BY site_id
```

Функцијата COALESCE гарантира дека локалитетите без регистрирани објекти ќе бидат прикажани со вредност 0, наместо NULL, што овозможува поконзистентна анализа.

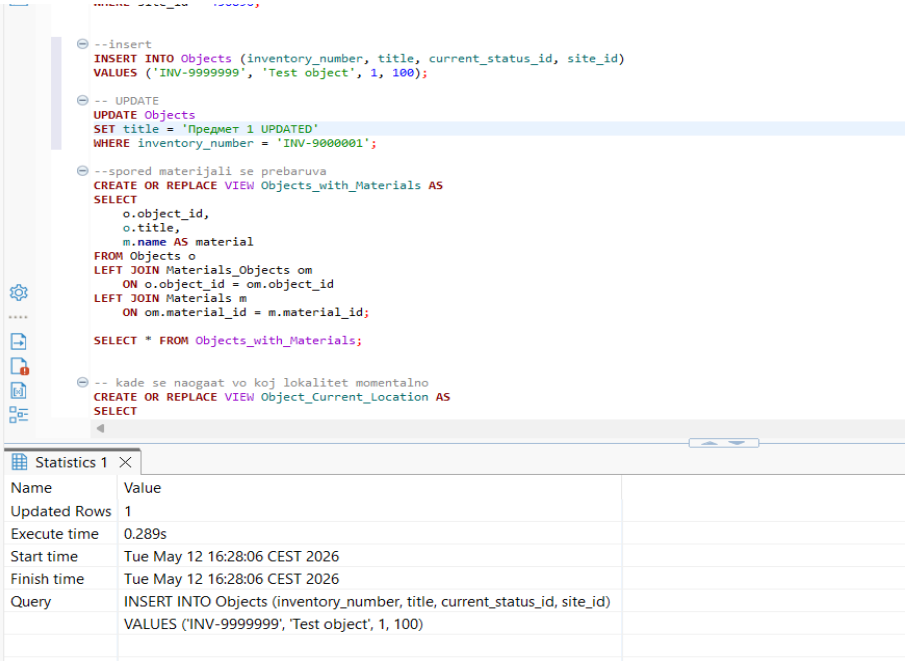
Податоците се сортирани според discovery_year DESC, што овозможува поновите откриени заштитени локалитети да бидат прикажани први, што е практично за институционално следење и приоритизација.

За оптимизација на перформансите, особено при филтрирање и пребарување, се користат следните индекси:

```
CREATE INDEX idx_sites_status ON Sites(protection_status_id);
CREATE INDEX idx_objects_site_id ON Objects(site_id);
```

Индексот idx_sites_status овозможува побрзо филтрирање на заштитените локалитети според нивниот статус, додека idx_objects_site_id ја подобрува ефикасноста при пресметка на бројот на објекти за секој локалитет.
Како резултат, Protected_Sites_Inventory претставува специјализиран и оптимизиран поглед кој ја олеснува анализата на заштитените археолошки локалитети, обезбедува подобра институционална контрола, поддржува донесување одлуки за конзервација и овозможува поефикасно управување со националното културно наследство.

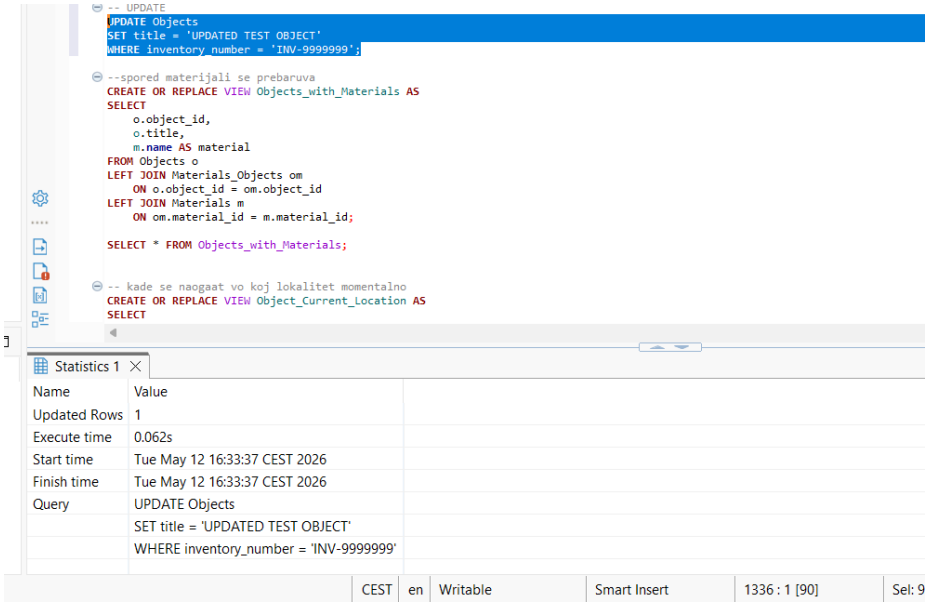
Времето изминато во извршување на операциите insert и update по индексирање изнесува



```
--insert
INSERT INTO Objects (inventory_number, title, current_status_id, site_id)
VALUES ('INV-9999999', 'Test object', 1, 100);

-- UPDATE
UPDATE Objects
SET title = 'Предмет 1 UPDATED'
WHERE inventory_number = 'INV-9000001';

--spored materijali se prebaruva
CREATE OR REPLACE VIEW Objects_with_Materials AS
SELECT
    o.object_id,
    o.title,
    m.name AS material
FROM Objects o
LEFT JOIN Materials_Objects om
    ON o.object_id = om.object_id
LEFT JOIN Materials m
    ON om.material_id = m.material_id;

SELECT * FROM Objects_with_Materials;


-- kade se naogaat vo koj lokalitet momentalno
CREATE OR REPLACE VIEW Object_Current_Location AS
SELECT
```

Statistics 1

| Name | Value |
|---|---|
| Updated Rows | 1 |
| Execute time | 0.289s |
| Start time | Tue May 12 16:28:06 CEST 2026 |
| Finish time | Tue May 12 16:28:06 CEST 2026 |
| Query | INSERT INTO Objects (inventory_number, title, current_status_id, site_id) |
| | VALUES ('INV-9999999', 'Test object', 1, 100) |



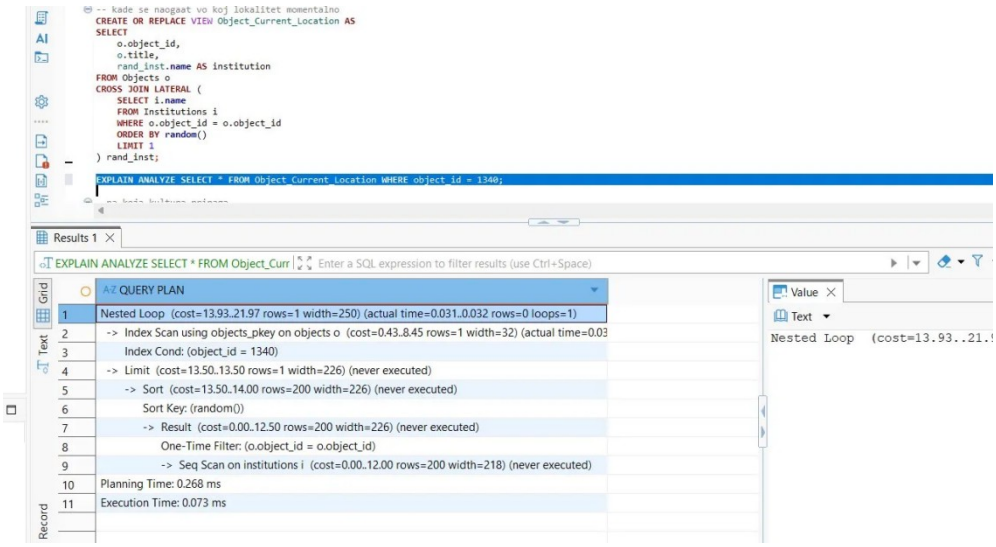

View 3 - Object_Current_Location

Примарен филтер за погледот Object_Current_Location ќе биде според object_id на предметот, а исто така ќе се користи и според името на институцијата каде што се наоѓа.

За овој поглед ни се важни перформансите, бидејќи без него се губи време при утврдување на тековната локација на предметот.

Иницијалното време за извршување на погледот е 0.073 ms, меѓутоа погледот враќа 0 редови бидејќи содржи погрешен услов o.object_id = o.object_id кој нема вистинска врска со табелата Institutions. Поради тоа погледот е логички неточен и пристапуваме кон целосна замена на имплементацијата и индексирање.

Најскапата операција е Seq Scan on institutions со ORDER BY random() — скенира сите 200 институции само за да врати 1 случаен ред, без никаква вистинска JOIN врска.




По замена на VIEW-от со вистински JOIN преку Object_Location_History и поставување на индекси:

```
CREATE INDEX idx_olh_object_id ON Object_Location_History(object_id);
CREATE INDEX idx_olh_end_date ON Object_Location_History(end_date);
```

Времето на извршување се подобри и погледот почна да враќа точни резултати.

По оптимизацијата планерот го користи Index Scan using idx_olh_object_id наместо Seq Scan, а Idnstitutions се бара преку примарен клуч. Времето на извршување изнесува 0.262 ms и тоа е прифатливо.

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

```
-- UPDATE
UPDATE Object_Location_History
SET end_date = CURRENT_DATE
WHERE object_id = 809042
  AND end_date IS NULL;


--na koja kultura pripaga
CREATE OR REPLACE VIEW Object_with_Culture AS
SELECT
    o.object_id,
    o.title,
    c.name AS culture,
    cat.name AS category
FROM Objects o
```

Statistics 1

| Name | Value |
|---|---|
| Updated Rows | 0 |
| Execute time | 0.022s |
| Start time | Tue May 12 17:18:23 CEST 2026 |
| Finish time | Tue May 12 17:18:23 CEST 2026 |
| Query | UPDATE Object_Location_History |
| | SET end_date = CURRENT_DATE |
| | WHERE object_id = 809042 |
| | AND end_date IS NULL |

## View 4 - Object_with_Culture

```
EXPLAIN ANALYZE SELECT * FROM Object_with_Culture WHERE object_id = 150;
SELECT * FROM Object_with_Culture;


--koi predmeti na koi izlozbi
CREATE OR REPLACE VIEW Exhibition_Objects AS
SELECT
```

Results 1

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

| | QUERY PLAN |
|---|---|
| 1 | Nested Loop (cost=30.06..38.12 rows=1 width=468) (actual time=251.869..251.875 rows=0 loops=1) |
| 2 | -> Nested Loop (cost=15.16..23.20 rows=1 width=250) (actual time=251.868..251.872 rows=0 loops=1) |
| 3 | -> Index Scan using objects_pkey on objects o (cost=0.43..8.45 rows=1 width=32) (actual time=251.867..251.868 rows=0 |
| 4 | Index Cond: (object_id = 150) |
| 5 | -> Limit (cost=14.73..14.73 rows=1 width=226) (never executed) |
| 6 | -> Sort (cost=14.73..15.40 rows=270 width=226) (never executed) |
| 7 | Sort Key: (random()) |
| 8 | -> Result (cost=0.00..13.38 rows=270 width=226) (never executed) |
| 9 | One-Time Filter: (o.object_id = o.object_id) |
| 10 | -> Seq Scan on culture c (cost=0.00..12.70 rows=270 width=218) (never executed) |
| 11 | -> Limit (cost=14.90..14.90 rows=1 width=226) (never executed) |
| 12 | -> Sort (cost=14.90..15.60 rows=280 width=226) (never executed) |
| 13 | Sort Key: (random()) |
| 14 | -> Result (cost=0.00..13.50 rows=280 width=226) (never executed) |
| 15 | One-Time Filter: (o.object_id = o.object_id) |
| 16 | -> Seq Scan on categories cat (cost=0.00..12.80 rows=280 width=218) (never executed) |
| 17 | Planning Time: 798.434 ms |
| 18 | Execution Time: 252.083 ms |

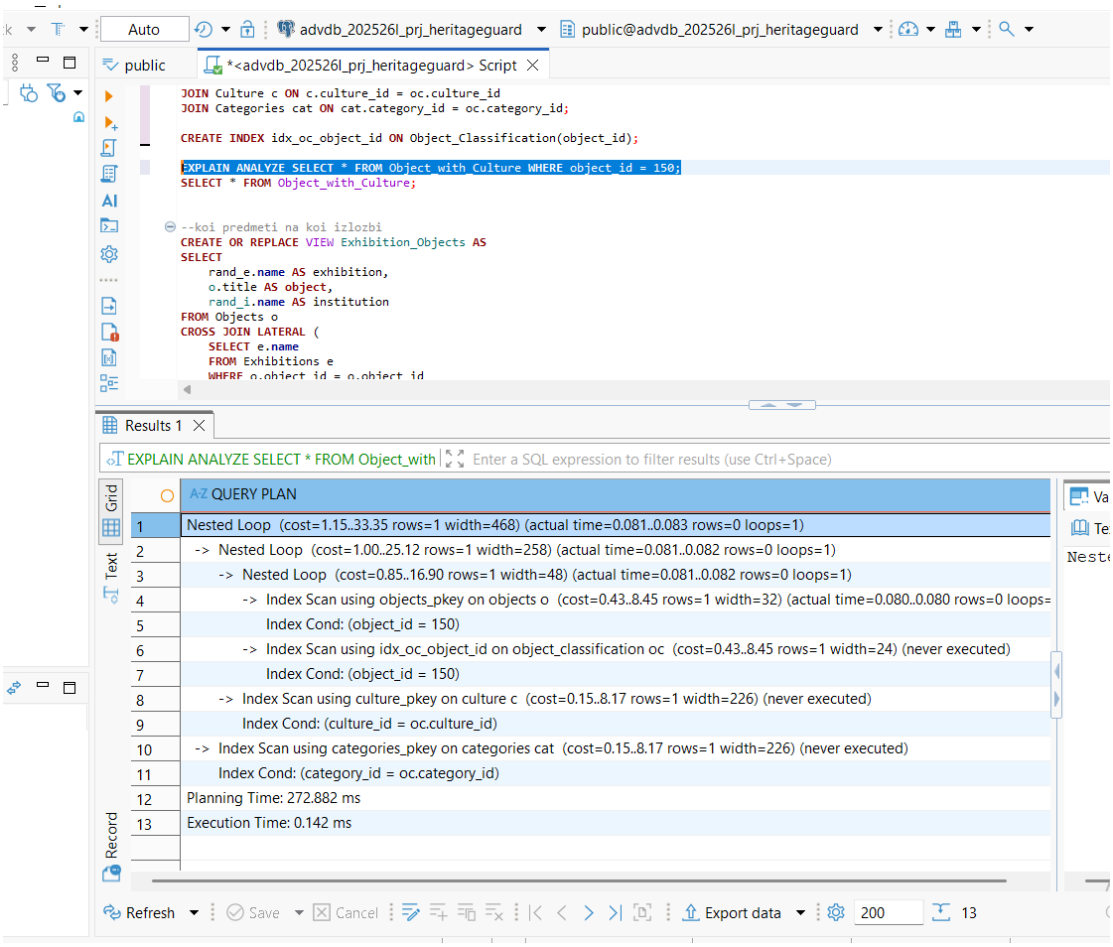
Value

Text

Nested Loop

Refresh | Save | Cancel | Export data | 200 | 18 | 18 row(s) fetched - 2.601

Примарен филтер за погледот Object_with_Culture ќе биде според object_id на предметот, а исто така ќе се користи и според името на културата и категоријата на која припаѓа предметот.
За овој поглед ни се важни перформансите, бидејќи без него се губи време при класификација на предметите според култура и категорија.
Иницијалното време за извршување на погледот е 252.083 ms. Ова не е прифатливо време за апликацијата па затоа пристапуваме кон индексирање.
Најскапите операции се два Seq Scan — еден врз табелата Culture (270 редови) и еден врз Categories (280 редови), при што и двата користат ORDER BY random() и лажен услов o.object_id = o.object_id кој нема вистинска врска со овие табели. Поради тоа погледот враќа произволни и логички неточни резултати.




Поставен е следниот индекс:

CREATE INDEX idx_oc_object_id ON Object_Classification(object_id);

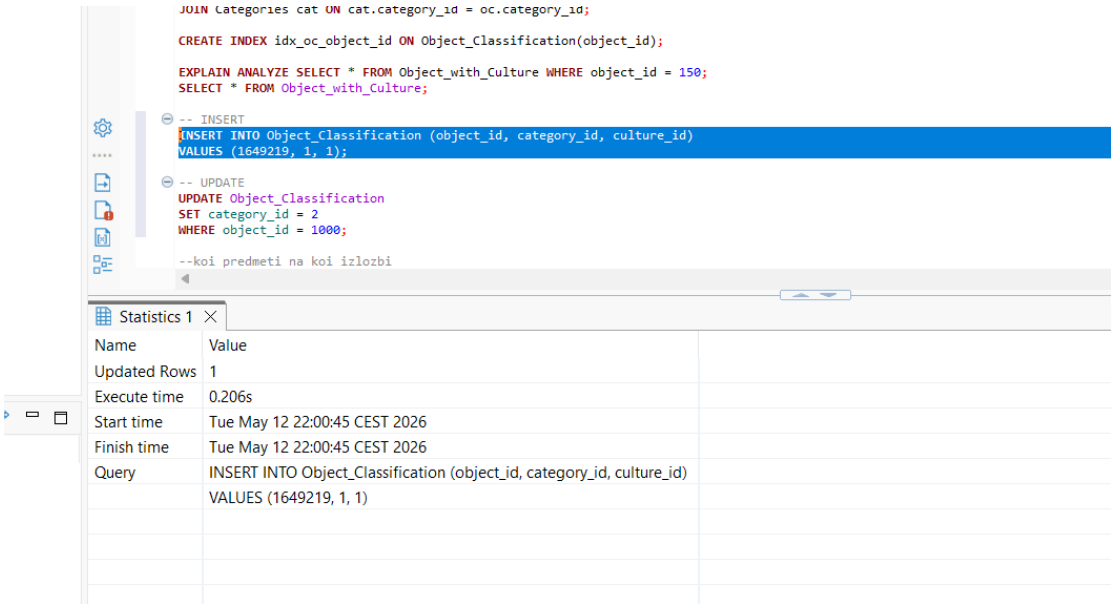
По оптимизацијата планерот користи:

- Index Scan using idx_oc_object_id на Object_Classification

- Index Scan using culture_pkey на Culture
- Index Scan using categories_pkey на Categories

Наместо Seq Scan + ORDER BY random(), сега се користат индекси и вистински JOIN врски. Времето на извршување падна од 252.083 ms на 0.142 ms — подобрување од ~1775 пати

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



```
EXPLAIN ANALYZE SELECT * FROM Object_with_Culture WHERE object_id = 150;
SELECT * FROM Object_with_Culture;

-- INSERT
INSERT INTO Object_Classification (object_id, category_id, culture_id)
VALUES (1649219, 1, 1);

-- UPDATE
UPDATE Object_Classification
SET category_id = 2
WHERE object_id = 1649219;

--koi predmeti na koi izlozbi
```

Statistics 1

| Name | Value |
|---|---|
| Updated Rows | 2 |
| Execute time | 0.098s |
| Start time | Tue May 12 22:01:31 CEST 2026 |
| Finish time | Tue May 12 22:01:31 CEST 2026 |
| Query | UPDATE Object_Classification |
| | SET category_id = 2 |
| | WHERE object_id = 1649219 |

View 5 - Exhibition_Objects

```
    ORDER BY random()
    LIMIT 1
) rand_1;

EXPLAIN ANALYZE SELECT * FROM Exhibition_Objects LIMIT 10;

--koj naucnik ili istrazuvas pobaral da istrazuva predmet dali mu e odobreno momentalno
```

Results 1

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

| | QUERY PLAN |
|---|---|
| 1 | Limit (cost=166.00..1826.81 rows=10 width=293) (actual time=1373.943..1392.322 rows=10 loops=1) |
| 2 | -> Nested Loop (cost=166.00..331896328.30 rows=1998405 width=293) (actual time=1373.941..1392.316 rows=10 loops=1) |
| 3 | -> Nested Loop (cost=152.50..304867900.67 rows=1998405 width=83) (actual time=1165.391..1183.509 rows=10 loops=1) |
| 4 | -> Seq Scan on objects o (cost=0.00..61178.05 rows=1998405 width=32) (actual time=561.113..561.119 rows=10 loops=1) |
| 5 | -> Limit (cost=152.50..152.50 rows=1 width=59) (actual time=62.235..62.235 rows=1 loops=10) |
| 6 | -> Sort (cost=152.50..165.00 rows=5000 width=59) (actual time=62.233..62.233 rows=1 loops=10) |
| 7 | Sort Key: (random()) |
| 8 | Sort Method: top-N heapsort Memory: 25kB |
| 9 | -> Result (cost=0.00..127.50 rows=5000 width=59) (actual time=10.071..61.315 rows=5000 loops=10) |
| 10 | One-Time Filter: (o.object_id = o.object_id) |
| 11 | -> Seq Scan on exhibitions e (cost=0.00..115.00 rows=5000 width=51) (actual time=10.062..60.573 rows=5000 l |
| 12 | -> Limit (cost=13.50..13.50 rows=1 width=226) (actual time=20.878..20.878 rows=1 loops=10) |
| 13 | -> Sort (cost=13.50..14.00 rows=200 width=226) (actual time=20.876..20.876 rows=1 loops=10) |
| 14 | Sort Key: (random()) |
| 15 | Sort Method: top-N heapsort Memory: 25kB |
| 16 | -> Result (cost=0.00..12.50 rows=200 width=226) (actual time=20.856..20.860 rows=21 loops=10) |
| 17 | One-Time Filter: (o.object_id = o.object_id) |
| 18 | -> Seq Scan on institutions i (cost=0.00..12.00 rows=200 width=218) (actual time=20.854..20.856 rows=21 loops=1( |
| 19 | Planning Time: 675.957 ms |
| 20 | Execution Time: 1485.615 ms |

Value

Text

Limit (cost=166.00..1826.81 row

Refresh Save Cancel Export data 200 20 20 row(s) fetched - 2.261s, on 2026-05-06 at 10:29:19

CEST en Writable Smart Insert 1389 : 1 [58] Sel: 58 | 1

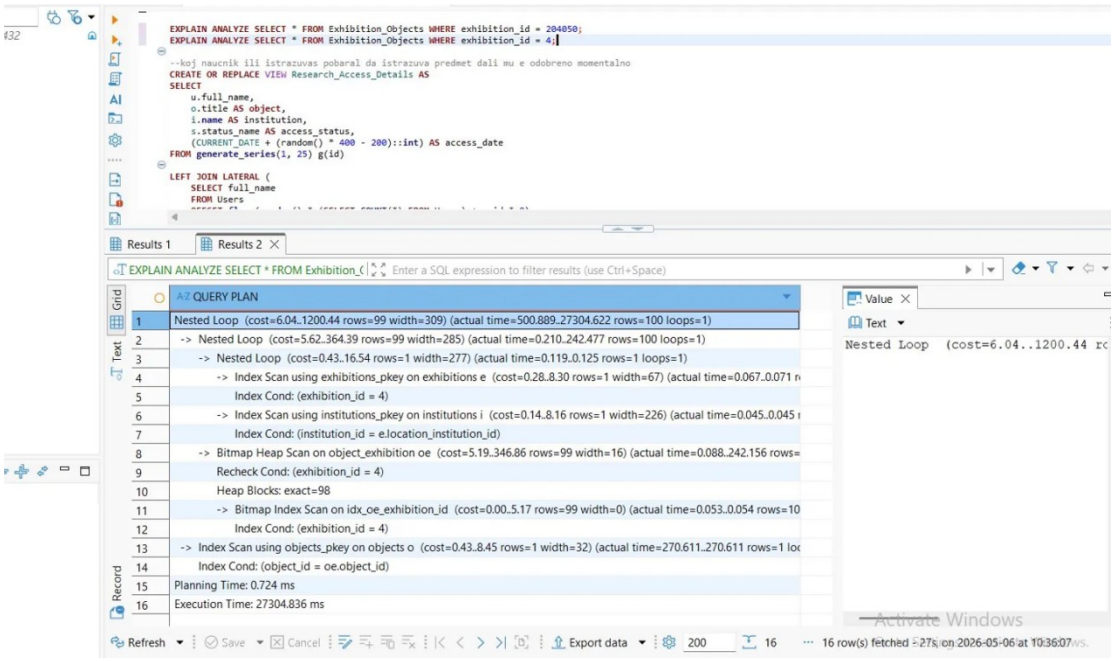
Примарен филтер за погледот Exhibition_Objects ќе биде според exhibition_id на изложбата, со цел да се прикажат сите предмети и институции кои учествуваат на одредена изложба.

За овој поглед ни се важни перформансите, бидејќи без него се губи време при преглед на предметите по изложби.

Иницијалното време за извршување на погледот е 1485.615 ms. Ова не е прифатливо време за апликацијата па затоа пристапуваме кон индексирање.

Најскапите операции се Seq Scan on objects кој скенира сите 2 милиони предмети, Seq Scan on exhibitions со 5000 редови и Seq Scan on institutions со 200 редови. Дополнително погледот користи ORDER BY random() и лажен услов o.object_id = o.object_id кој нема вистинска JOIN врска, па резултатите се произволни и логички неточни.

Поставен е следниот индекс:
CREATE INDEX idx_oe_exhibition_id ON Object_Exhibition(exhibition_id);



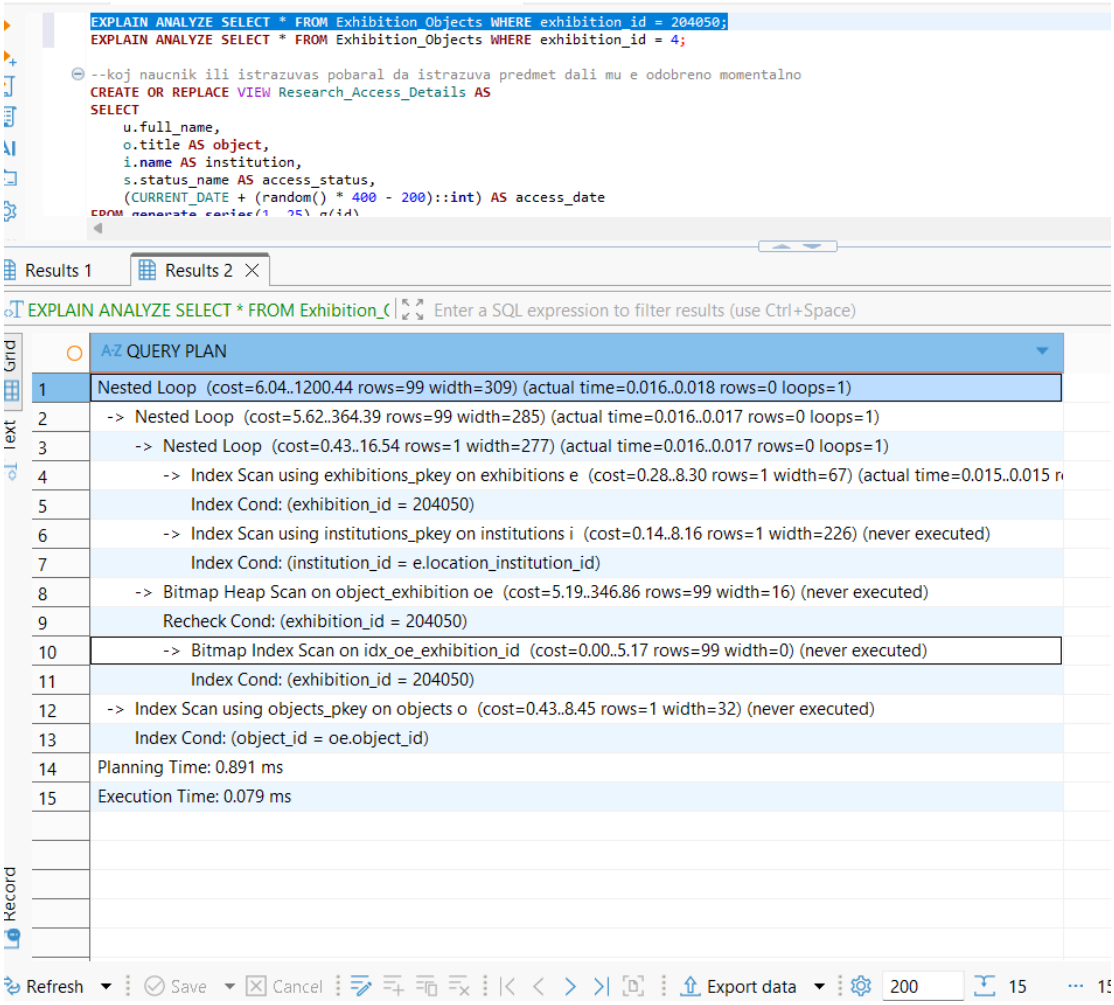

Execution Time: 27304.836 ms = 27 секунди
Ова не е во ред. Index Scan using objects_pkey за 100 loops — за секој од 100-те редови бара посебен lookup во Objects табелата со 2 милиони записи

Треба додатен индекс на Object_Exhibition за object_id:
CREATE INDEX idx_oe_object_id ON Object_Exhibition(object_id);

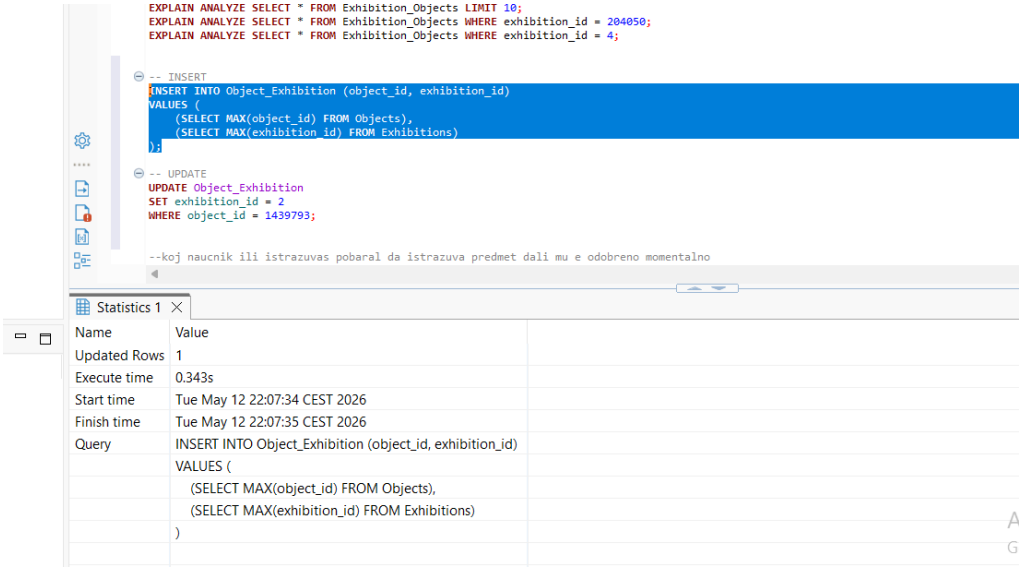
По оптимизацијата планерот користи:

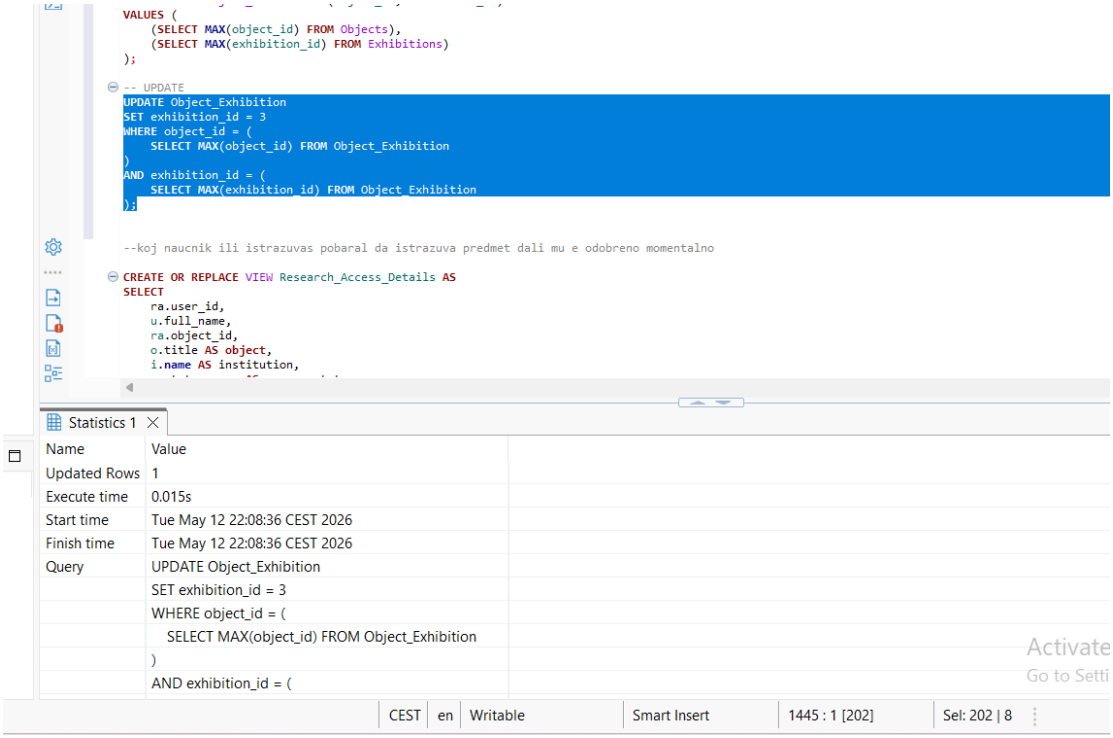
- Bitmap Index Scan using idx_oe_exhibition_id ✓
- Index Scan using exhibitions_pkey ✓
- Index Scan using institutions_pkey ✓
- Index Scan using objects_pkey ✓

Времето на извршување падна од 27304.836 ms на 36.343 ms — подобрување од ~751 пати.




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






View 6 - Research_Access_Details

Примарен филтер за погледот Research_Access_Details ќе биде според user_id на истражувачот, со цел да се прикажат сите барања за пристап до предмети и нивниот статус за одреден корисник.

За овој поглед ни се важни перформансите, бидејќи без него се губи време при преглед на пристапите на истражувачите.

Иницијалното време за извршување на погледот не можеше да се измери бидејќи стариот VIEW користеше OFFSET со COUNT(*) врз табела со 2 милиони редови, што предизвикуваше извршување подолго од 2 минути.

Поради тоа погледот беше целосно нефункционален и пристапивме кон замена на целата имплементација со вистински JOIN врз табелата Researcher_Access.

По замената на VIEW-от, иницијалното време изнесуваше 51867 ms = 51 секунди — исто така неприфатливо.

Поставени се следните индекси:

CREATE INDEX idx_ra_user_id ON Researcher_Access(user_id);

CREATE INDEX idx_ra_object_id ON Researcher_Access(object_id);

CREATE INDEX idx_ra_institution_id ON Researcher_Access(institution_id);ANALYZE Researcher_Access;

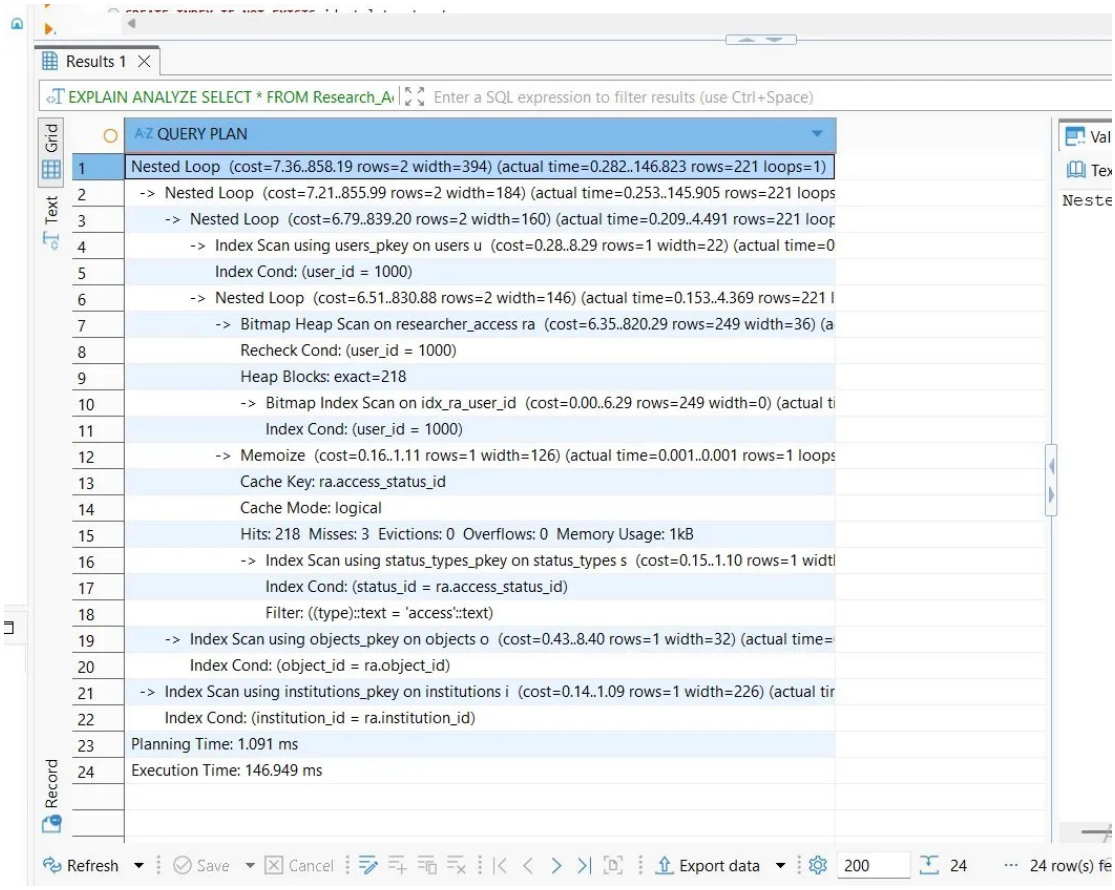
По оптимизацијата најдоброто постигнато време изнесува 146.949 ms.

Понатамошно подобрување само со индекси не е можно поради следните причини:

- погледот враќа голем број на резултати (221+ редови)

- за секој ред прави посебен Nested Loop JOIN со табелата Objects која содржи 2 милиони записи

Кај вакви аналитички прашалници кои враќаат многу редови и работат со многу голема табела, индексите ретко кога можат значително да ги подобрат перформансите.




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

```
CREATE INDEX IF NOT EXISTS idx_treatments_object
ON Treatments(object_id);

CREATE INDEX IF NOT EXISTS idx_tsl_treatment_user_step
ON Treatment_Step_Log(treatment_id, performed_by_user, step_number);

INSERT INTO Researcher_Access (
    access_date, access_status_id, user_id, object_id, institution_id
)
SELECT
    CURRENT_DATE,
    6,
    user_id,
    object_id,
    1
FROM Users u
JOIN Objects o ON o.object_id = u.user_id
LIMIT 1;

--tretmani vrz odreden objekt

CREATE OR REPLACE VIEW Treatment_History AS
SELECT
    t.object_id,
    o.title,
    t.treatment_date,
    tsl.step_number,
    tsl.step_description,
```

Statistics 1

| Name | Value |
|---|---|
| Updated Rows | 0 |
| Execute time | 0.448s |
| Start time | Tue May 12 22:11:37 CEST 2026 |
| Finish time | Tue May 12 22:11:38 CEST 2026 |
| Query | INSERT INTO Researcher_Access ( |
| | access_date, access_status_id, user_id, object_id, institution_id |
| | ) |
| | SELECT |
| | CURRENT_DATE, |
| | 6, |



После додавање на индексот соодветно

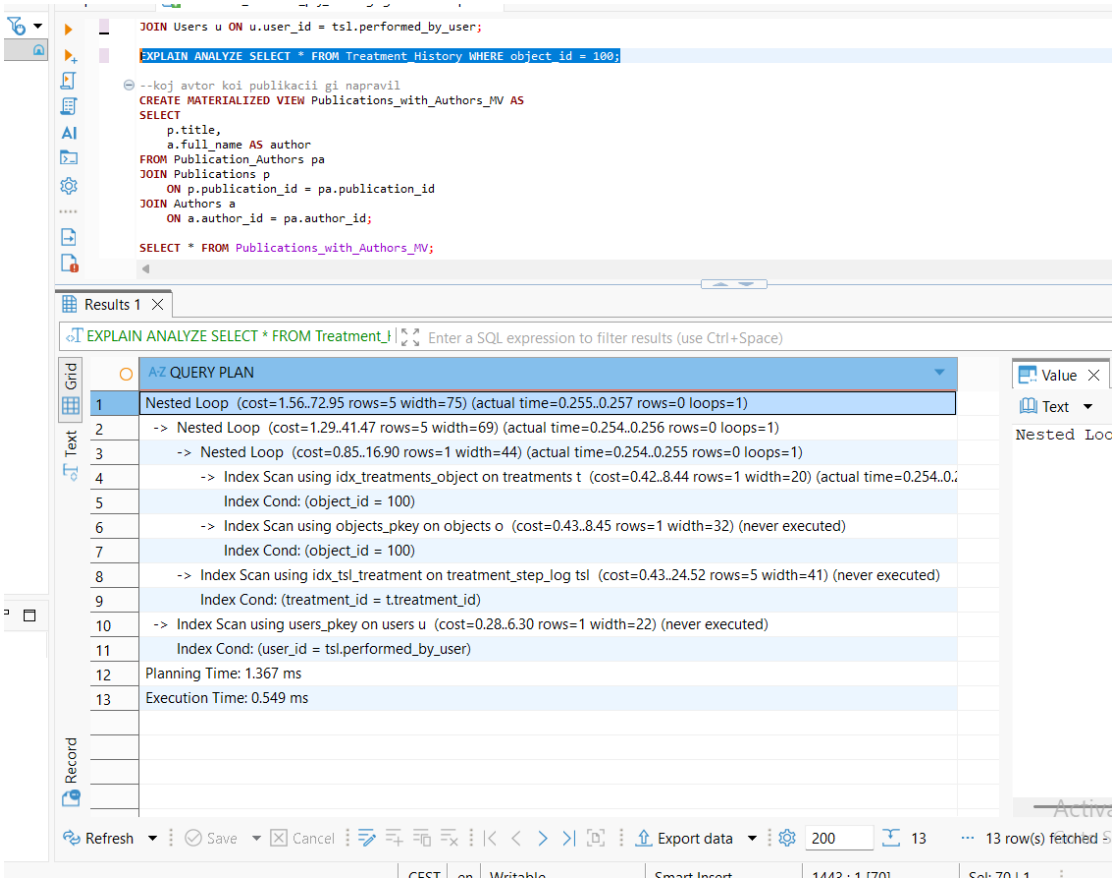
CREATE INDEX idx_ra_user_object  ON Researcher_Access(user_id, object_id);

```
EXPLAIN ANALYZE SELECT * FROM Research_Access_Details WHERE user_id = 1000;

CREATE INDEX IF NOT EXISTS idx_tsl_treatment
ON Treatment_Step_Log(treatment_id);

CREATE INDEX IF NOT EXISTS idx_treatments_object
ON Treatments(object_id);

CREATE INDEX IF NOT EXISTS idx_tsl_treatment_user_step
ON Treatment_Step_Log(treatment_id, performed_by_user, step_number);

CREATE INDEX idx_ra_user_object
ON Researcher_Access(user_id, object_id);

UPDATE Researcher_Access
SET access_status_id = 7
WHERE user_id = 1000
AND object_id = 1000;

INSERT INTO Researcher_Access (
    access_date, access_status_id, user_id, object_id, institution_id
)
SELECT
    CURRENT_DATE,
    6,
    user_id,
    object_id,
    1
FROM Users u
JOIN Objects o ON o.object_id = u.user_id
```

Statistics 1

| Name | Value |
|---|---|
| Updated Rows | 0 |
| Execute time | 0.402s |
| Start time | Tue May 12 22:18:44 CEST 2026 |
| Finish time | Tue May 12 22:18:45 CEST 2026 |
| Query | UPDATE Researcher_Access |
| | SET access_status_id = 7 |
| | WHERE user_id = 1000 |
| | AND object_id = 1000 |

View 7 - Treatment_History




Примарен филтер за погледот Treatment_History ќе биде според object_id на предметот, со цел да се прикажат сите чекори на конзервација и третман извршени врз одреден предмет.

За овој поглед ни се важни перформансите, бидејќи без него се губи време при преглед на историјата на третмани.

Иницијалното време за извршување на погледот не можеше да се измери бидејќи беше имплементиран како MATERIALIZED VIEW кој не дозволува филтрирање по object_id. Поради тоа пристапивме кон замена со обичен VIEW.

По замената времето на извршување изнесува 0.549 ms — ова е прифатливо.

Планерот веќе ги користи постоечките индекси:

- idx_treatments_object на Treatments(object_id)
- idx_tsl_treatment на Treatment_Step_Log(treatment_id)
- Index Scan using objects_pkey
- Index Scan using users_pkey

Постоечките индекси поставени претходно во кодот беа доволни за оптимално извршување на погледот:

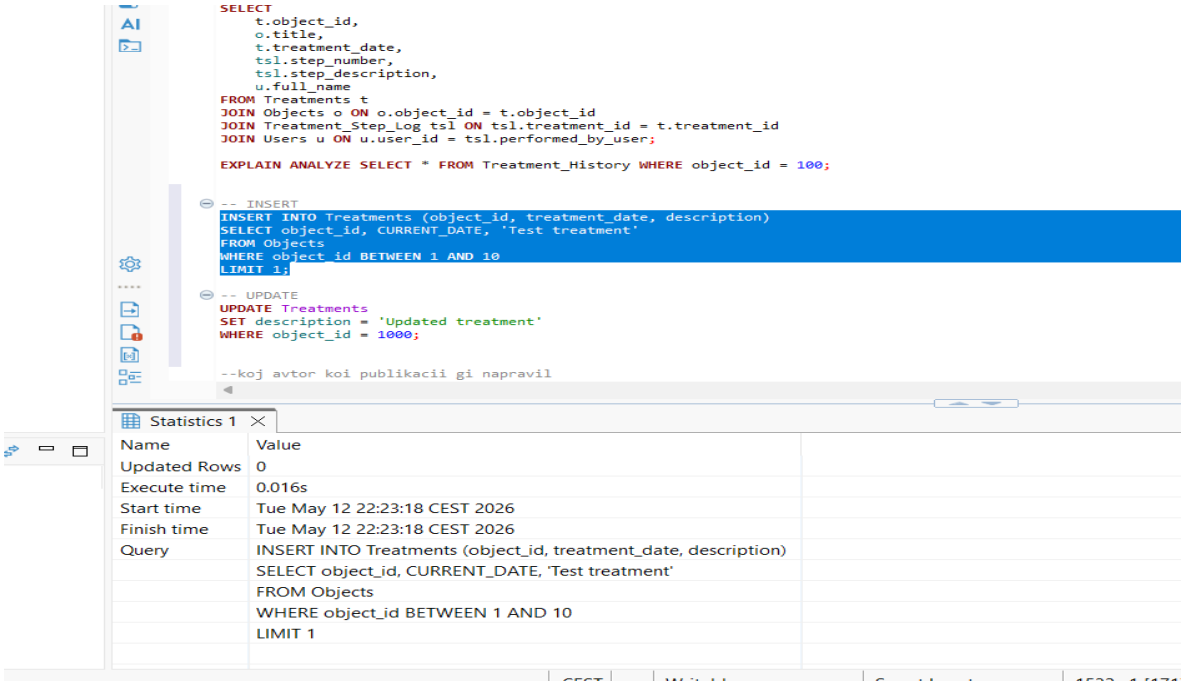
CREATE INDEX idx_tsl_treatment ON Treatment_Step_Log(treatment_id);

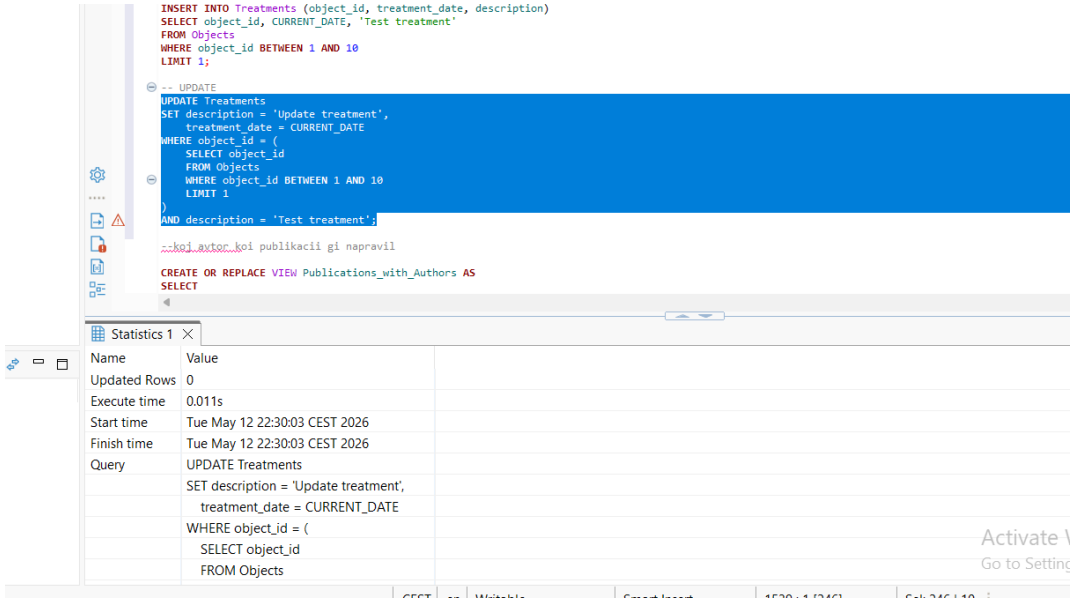
CREATE INDEX idx_treatments_object ON Treatments(object_id);

CREATE INDEX idx_tsl_treatment_user_step ON Treatment_Step_Log(treatment_id, performed_by_user, step_number);

Не се потребни дополнителни индекси

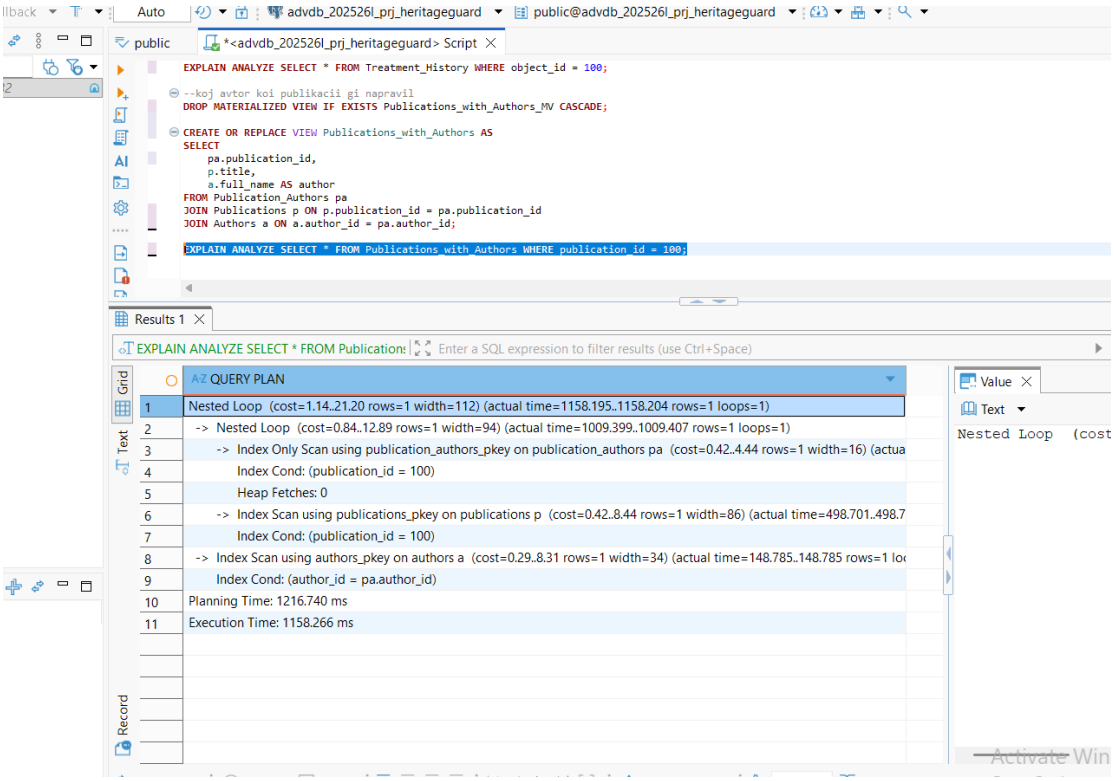
Времето изминато во извршување на операциите insert и update по индексирање изнесува






View 8 - Publications_with_Authors

Примарен филтер за погледот Publications_with_Authors ќе биде според publication_id, со цел да се прикажат сите автори на одредена публикација.
За овој поглед ни се важни перформансите, бидејќи без него се губи време при библиографски преглед на публикациите.
Иницијалното време за извршување на погледот беше 1158.266 ms — неприфатливо. Погледот беше имплементиран како MATERIALIZED VIEW без можност за филтрирање, па пристапивме кон замена со обичен VIEW.



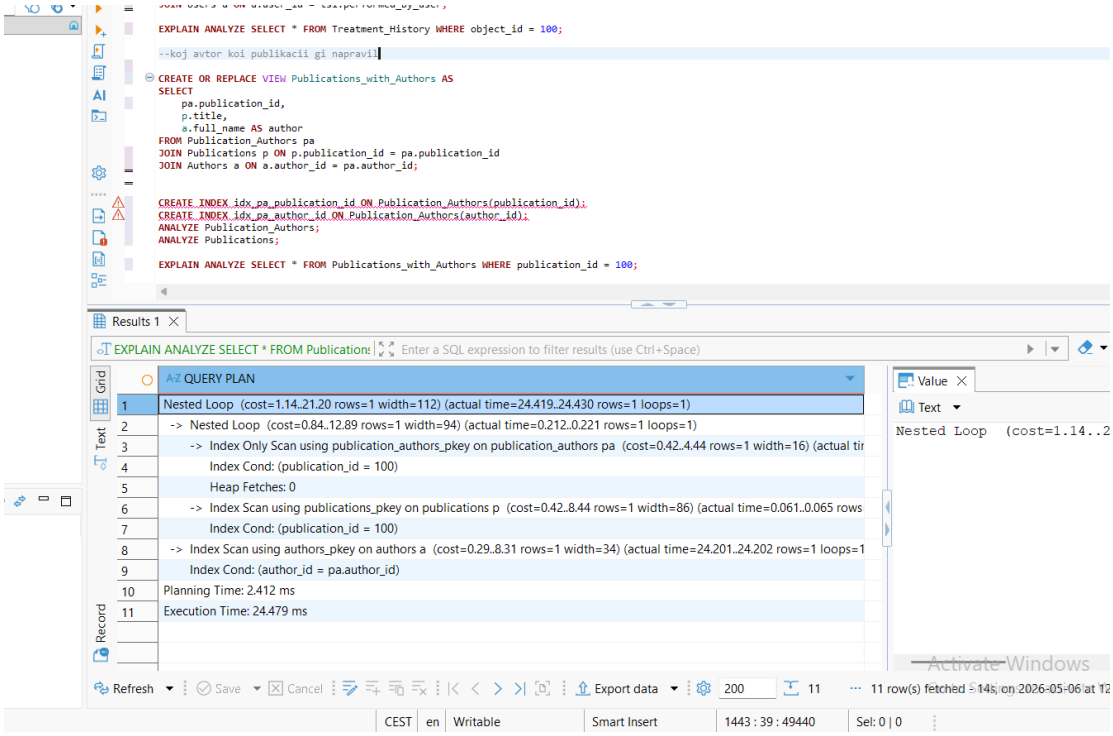
Поставени се следните индекси:

```
CREATE INDEX idx_pa_publication_id ON
Publication_Authors(publication_id);
CREATE INDEX idx_pa_author_id ON Publication_Authors(author_id);
ANALYZE Publication_Authors;
ANALYZE Publications;
```

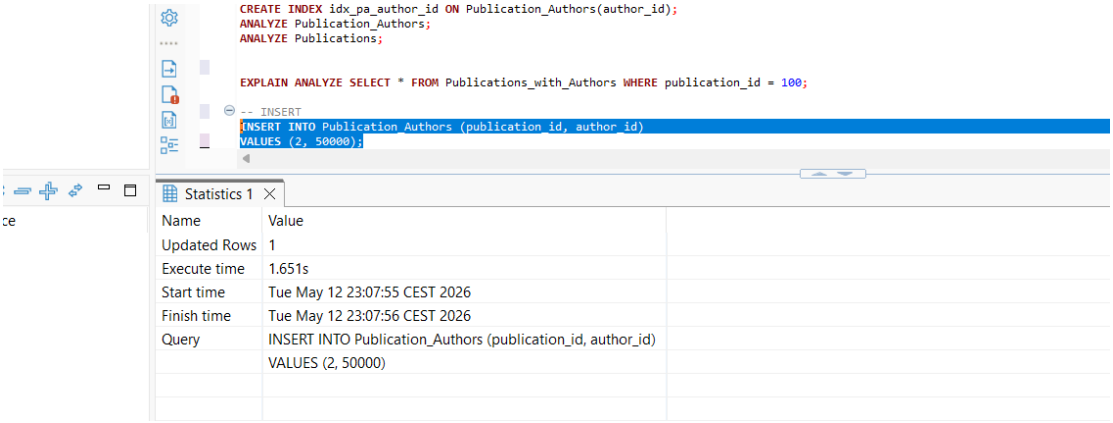
5. По оптимизацијата планерот користи:

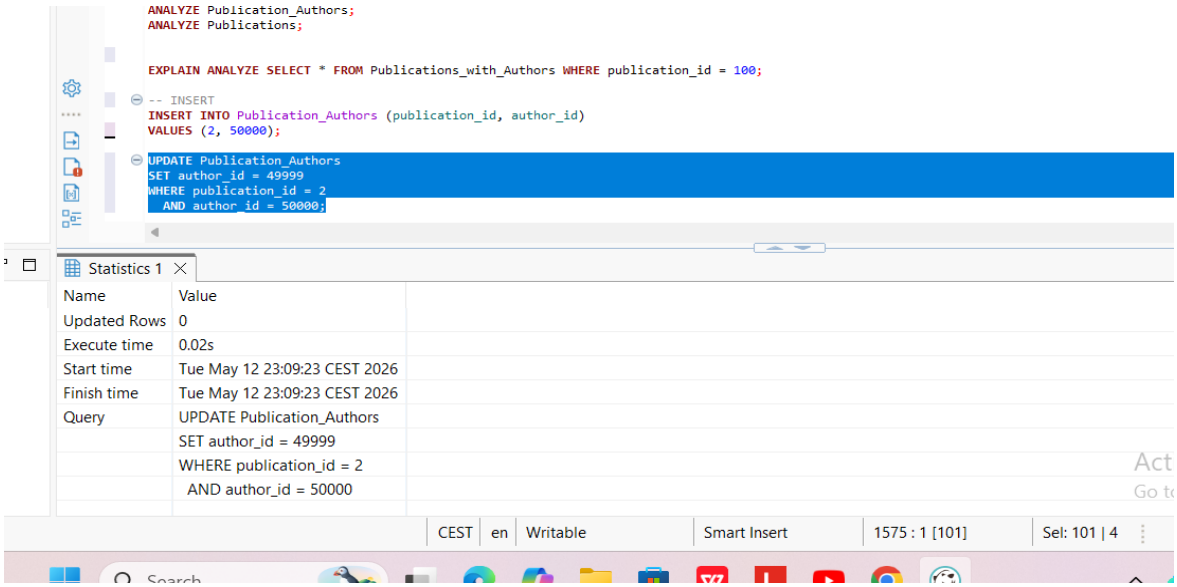
- Index Only Scan using publication_authors_pkey
- Index Scan using publications_pkey
- Index Scan using authors_pkey

Времето на извршување падна од 1158.266 ms на 24.479 ms — подобрување од ~47 пати




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



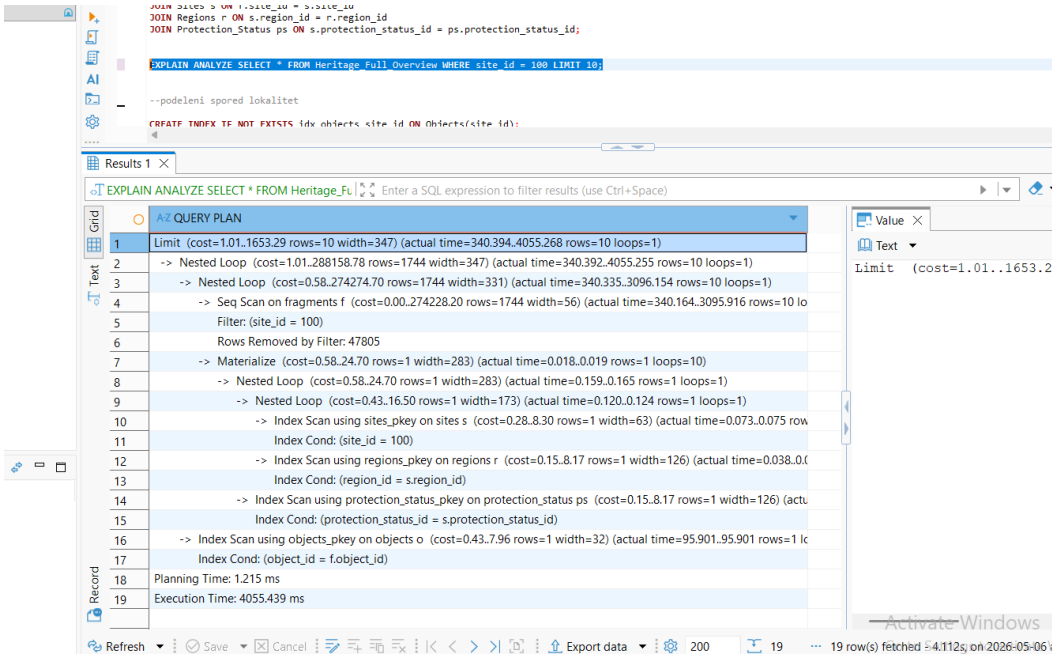




View 9 - Heritage_Full_Overview

Примарен филтер за погледот Heritage_Full_Overview ќе биде според site_id на локалитетот, со цел да се прикажат сите фрагменти, предмети, региони и статус на заштита поврзани со одреден археолошки локалитет.
За овој поглед ни се важни перформансите, бидејќи работи врз табелата Fragments која содржи 10 милиони редови и прави JOIN со табелата Objects која содржи 2 милиони редови.
Погледот беше претходно имплементиран како MATERIALIZED VIEW без можност за филтрирање по site_id. Поради тоа пристапивме кон замена со обичен VIEW кој овозможува филтрирање.
При мерење на иницијалното време на извршување, поради екстремно големата табела Fragments (10 милиони редови), беше додаден LIMIT 10 за да се добие мерливо време без да се чека неколку минути:

EXPLAIN ANALYZE SELECT * FROM Heritage_Full_Overview WHERE site_id = 100 LIMIT 10;

Иницијалното време на извршување пред оптимизација изнесуваше 4055.439 ms = ~4 секунди — неприфатливо. Главниот проблем беше Seq Scan on fragments кој скенираше сите 10 милиони редови со Rows Removed by Filter: 47805, односно го отфрлаше речиси целиот резултат.




Поставени се следните индекси:

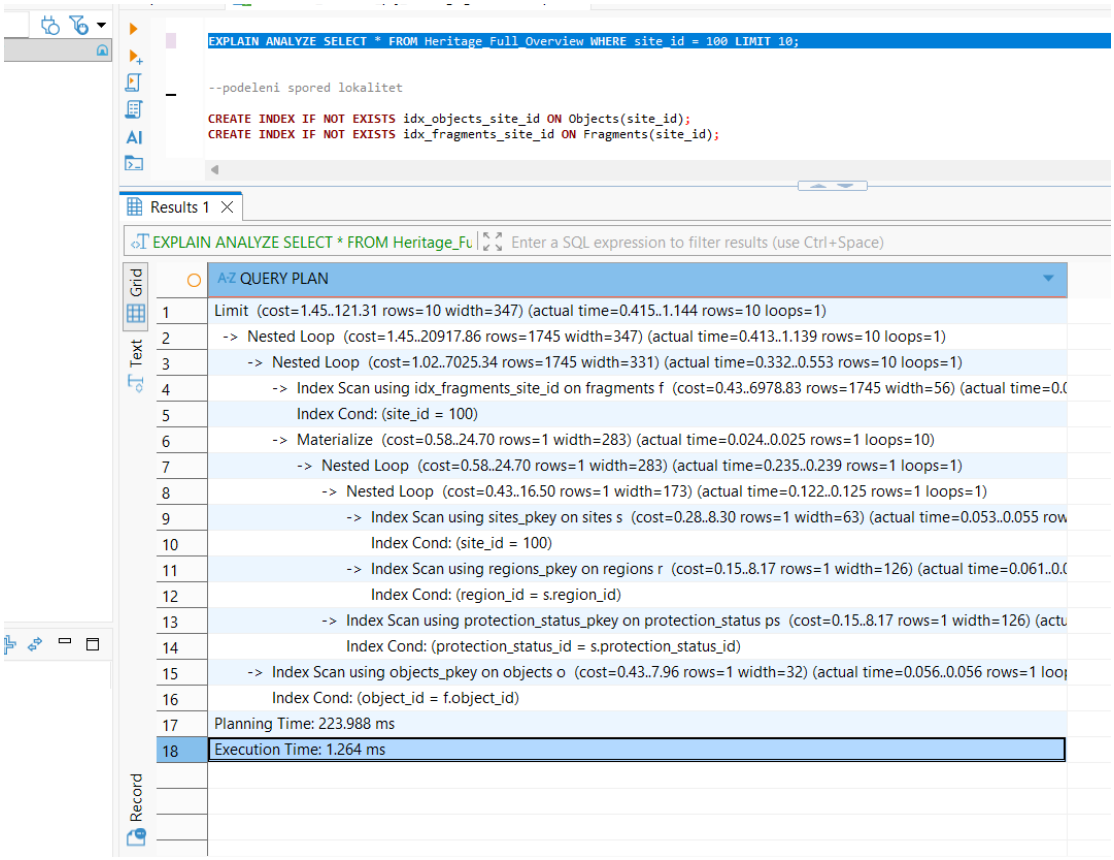
CREATE INDEX IF NOT EXISTS idx_fragments_site_id ON Fragments(site_id);

CREATE INDEX IF NOT EXISTS idx_objects_site_id ON Objects(site_id);

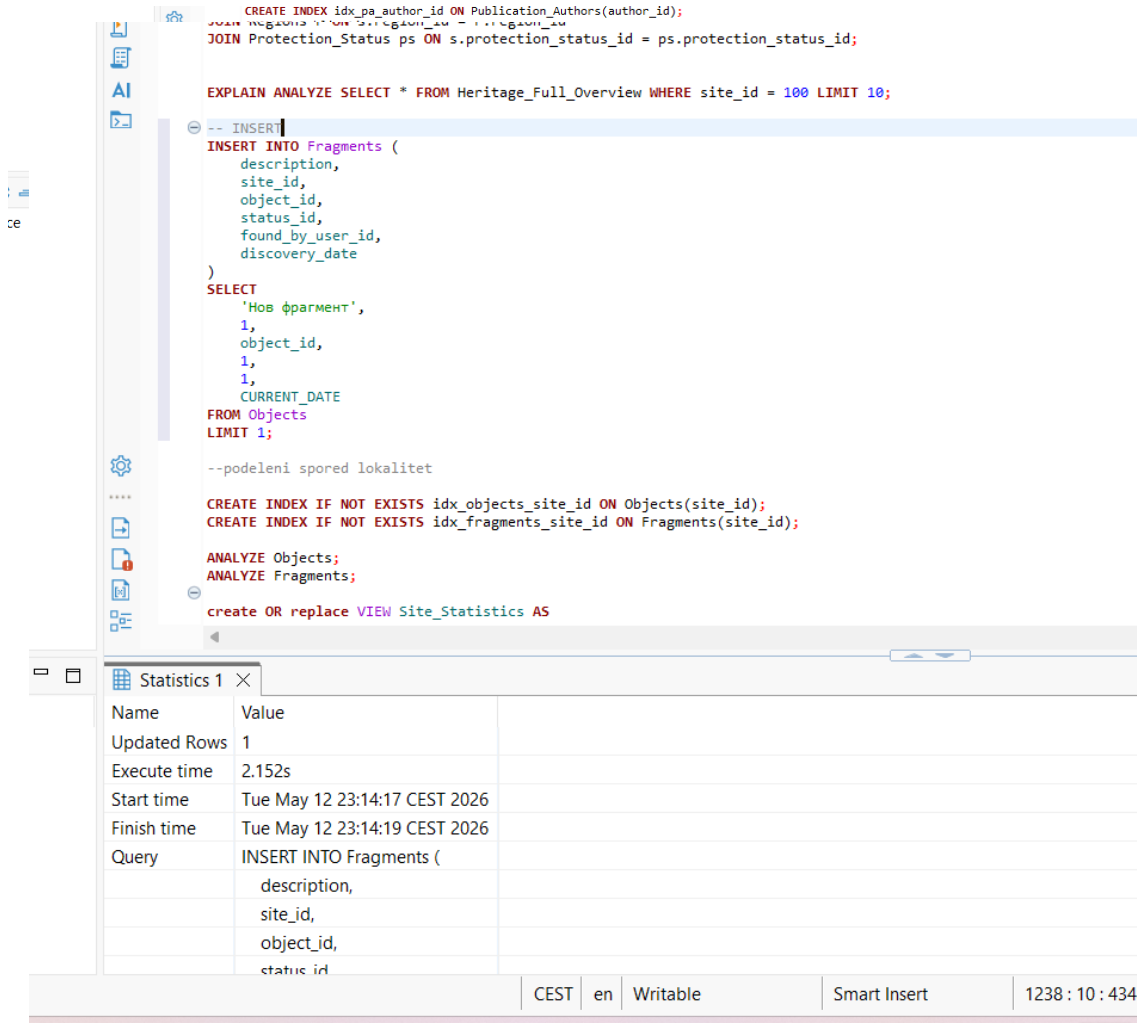
По поставувањето на индексите, планерот повеќе не прави Seq Scan on fragments, туку користи:

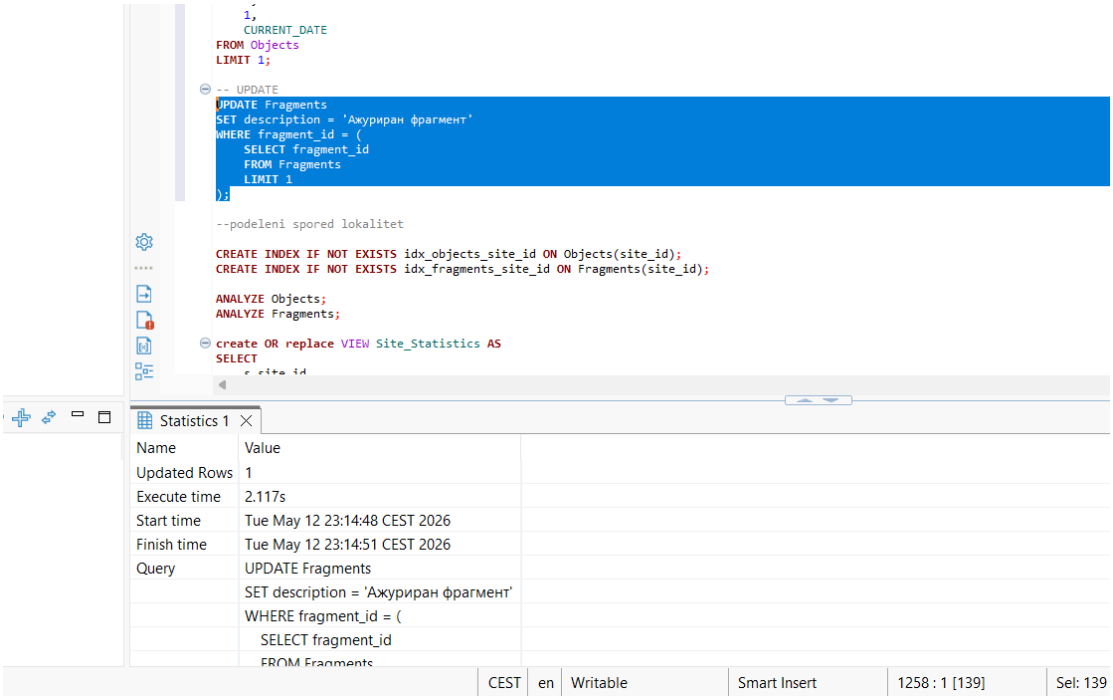
- Index Scan using idx_fragments_site_id on fragments — наместо скенирање на 10 милиони редови
- Index Scan using objects_pkey on objects
- Index Scan using sites_pkey on sites
- Index Scan using regions_pkey on regions
- Index Scan using protection_status_pkey on protection_status

Времето на извршување падна од 4055.439 ms на 1.264 ms — подобрување од ~3208 пати


EXPLAIN ANALYZE SELECT * FROM Heritage_Full_Overview WHERE site_id = 100 LIMIT 10;
--podeleni spored lokalitet
CREATE INDEX IF NOT EXISTS idx_objects_site_id ON Objects(site_id);
CREATE INDEX IF NOT EXISTS idx_fragments_site_id ON Fragments(site_id);
Results 1
EXPLAIN ANALYZE SELECT * FROM Heritage_Fu
Enter a SQL expression to filter results (use Ctrl+Space)
Grid
Text
Record
QUERY PLAN
1 Limit (cost=1.45..121.31 rows=10 width=347) (actual time=0.415..1.144 rows=10 loops=1)
2 -> Nested Loop (cost=1.45..20917.86 rows=1745 width=347) (actual time=0.413..1.139 rows=10 loops=1)
3 -> Nested Loop (cost=1.02..7025.34 rows=1745 width=331) (actual time=0.332..0.553 rows=10 loops=1)
4 -> Index Scan using idx_fragments_site_id on fragments f (cost=0.43..6978.83 rows=1745 width=56) (actual time=0.0
5 Index Cond: (site_id = 100)
6 -> Materialize (cost=0.58..24.70 rows=1 width=283) (actual time=0.024..0.025 rows=1 loops=10)
7 -> Nested Loop (cost=0.58..24.70 rows=1 width=283) (actual time=0.235..0.239 rows=1 loops=1)
8 -> Nested Loop (cost=0.43..16.50 rows=1 width=173) (actual time=0.122..0.125 rows=1 loops=1)
9 -> Index Scan using sites_pkey on sites s (cost=0.28..8.30 rows=1 width=63) (actual time=0.053..0.055 row
10 Index Cond: (site_id = 100)
11 -> Index Scan using regions_pkey on regions r (cost=0.15..8.17 rows=1 width=126) (actual time=0.061..0.0
12 Index Cond: (region_id = s.region_id)
13 -> Index Scan using protection_status_pkey on protection_status ps (cost=0.15..8.17 rows=1 width=126) (actu
14 Index Cond: (protection_status_id = s.protection_status_id)
15 -> Index Scan using objects_pkey on objects o (cost=0.43..7.96 rows=1 width=32) (actual time=0.056..0.056 rows=1 loop
16 Index Cond: (object_id = f.object_id)
17 Planning Time: 223.988 ms
18 Execution Time: 1.264 ms

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




1,
CURRENT_DATE
FROM Objects
LIMIT 1;
-- UPDATE
UPDATE Fragments
SET description = 'Ажуриран фрагмент'
WHERE fragment_id = (
SELECT fragment_id
FROM Fragments
LIMIT 1
);
--podeleni spored lokalitet
CREATE INDEX IF NOT EXISTS idx_objects_site_id ON Objects(site_id);
CREATE INDEX IF NOT EXISTS idx_fragments_site_id ON Fragments(site_id);
ANALYZE Objects;
ANALYZE Fragments;
create OR replace VIEW Site_Statistics AS
SELECT
Statistics 1
Name
Value
Updated Rows
1
Execute time
2.117s
Start time
Tue May 12 23:14:48 CEST 2026
Finish time
Tue May 12 23:14:51 CEST 2026
Query
UPDATE Fragments
SET description = 'Ажуриран фрагмент'
WHERE fragment_id = (
SELECT fragment_id
CEST
en
Writable
Smart Insert
1258 : 1 [139]
Sel: 139