

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

The screenshot displays a database query editor with the following SQL query:

```
relname AS table_name,  
pg_size_pretty(pg_total_relation_size(releid)) AS total_size,  
pg_size_pretty(pg_relation_size(releid)) AS data_size,  
pg_size_pretty(pg_total_relation_size(releid) - pg_relation_size(releid)) AS index_size  
FROM pg_catalog.pg_statio_user_tables  
WHERE relname IN ('fragments', 'objects', 'sites');
```

Below the query, the execution plan for the query is shown:

```
explain analyze select * from site_statistics where site_id=39568;
```

The execution plan details the following steps:

1. Nested Loop Left Join (cost=1.29..70.94 rows=1 width=189) (actual time=0.136..0.138 rows=0 loops=1)
 - 2. -> Nested Loop Left Join (cost=0.86..27.58 rows=1 width=173) (actual time=0.136..0.137 rows=0 loops=1)
 - 3. -> Nested Loop (cost=0.43..16.50 rows=1 width=165) (actual time=0.135..0.136 rows=0 loops=1)
 - 4. -> Index Scan using sites_pkey on sites s (cost=0.28..8.30 rows=1 width=55) (actual time=0.135..0.135 rows=0 loops=1)
 - 5. Index Cond: (site_id = 39568)
 - 6. -> Index Scan using regions_pkey on regions r (cost=0.15..8.17 rows=1 width=126) (never executed)
 - 7. Index Cond: (region_id = s.region_id)
 - 8. -> GroupAggregate (cost=0.43..11.06 rows=1 width=16) (never executed)
 - 9. -> Index Only Scan using idx_objects_site_id on objects (cost=0.43..10.22 rows=331 width=8) (never executed)
 - 10. Index Cond: (site_id = 39568)
 - 11. Heap Fetches: 0
 - 12. -> GroupAggregate (cost=0.43..43.32 rows=1 width=16) (never executed)
 - 13. -> Index Only Scan using idx_fragments_site_id on fragments (cost=0.43..38.95 rows=1744 width=8) (never executed)
 - 14. Index Cond: (site_id = 39568)
 - 15. Heap Fetches: 0
 - 16. Planning Time: 0.603 ms
 - 17. Execution Time: 0.215 ms

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

The screenshot displays a PostgreSQL database interface with a SQL editor and a statistics window. The SQL editor contains the following queries:

```
-- 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;  
  
-- VIEW  
EXPLAIN ANALYZE  
SELECT * FROM Site_Statistics  
WHERE site_id = 100;  
  
--zastiteni lokaliteti  
CREATE OR REPLACE VIEW Protected_Sites_Inventory AS  
SELECT  
s.site_id,  
s.site_name,
```

The statistics window, titled 'Statistics 1', shows the following data:

Name	Value
Updated Rows	1
Execute time	0.269s
Start time	Tue May 12 15:53:34 CEST 2026
Finish time	Tue May 12 15:53:35 CEST 2026
Query	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)

The screenshot shows a database query editor with the following SQL commands:

```
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;
```

Below the query editor, a "Statistics 1" window is open, displaying the following data:

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

The screenshot shows a database query editor with the following SQL commands:

```
EXPLAIN ANALYZE
SELECT *
FROM Protected_Sites_Inventory
WHERE site_id = 456890;

--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
```

Below the query editor, an "EXPLAIN ANALYZE" window is open, displaying the query plan for the query: `EXPLAIN ANALYZE SELECT * FROM Protected_Sites_Inventory WHERE site_id = 456890;`

Line	Operation	Cost	Rows	Width	Actual Time	Actual Rows	Actual Loops
5	Nested Loop	0.58..24.68	rows=1	width=169	0.143..0.144	rows=0	loops=1
6	Join Filter	(s.protection_status_id = ps.protection_status_id)					
7	Nested Loop	0.43..16.50	rows=1	width=177	0.142..0.143	rows=0	loops=1
8	Index Scan using sites_pkey on sites s	0.28..8.30	rows=1	width=67	0.142..0.142	rows=0	loops=1
9	Index Cond	(site_id = 456890)					
10	Index Scan using regions_pkey on regions r	0.15..8.17	rows=1	width=126	(never executed)		
11	Index Cond	(region_id = s.region_id)					
12	Index Scan using protection_status_name_key on protection_status ps	0.15..8.17	rows=1	width=8	(never executed)		
13	Index Cond	((name)::text = 'Заштитен')::text					
14	GroupAggregate	0.43..11.06	rows=1	width=16	(never executed)		
15	Index Only Scan using idx_objects_site_id on objects	0.43..10.22	rows=331	width=8	(never executed)		
16	Index Cond	(site_id = 456890)					
17	Heap Fetches	0					
18	Planning Time	0.605 ms					
19	Execution Time	0.225 ms					

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.

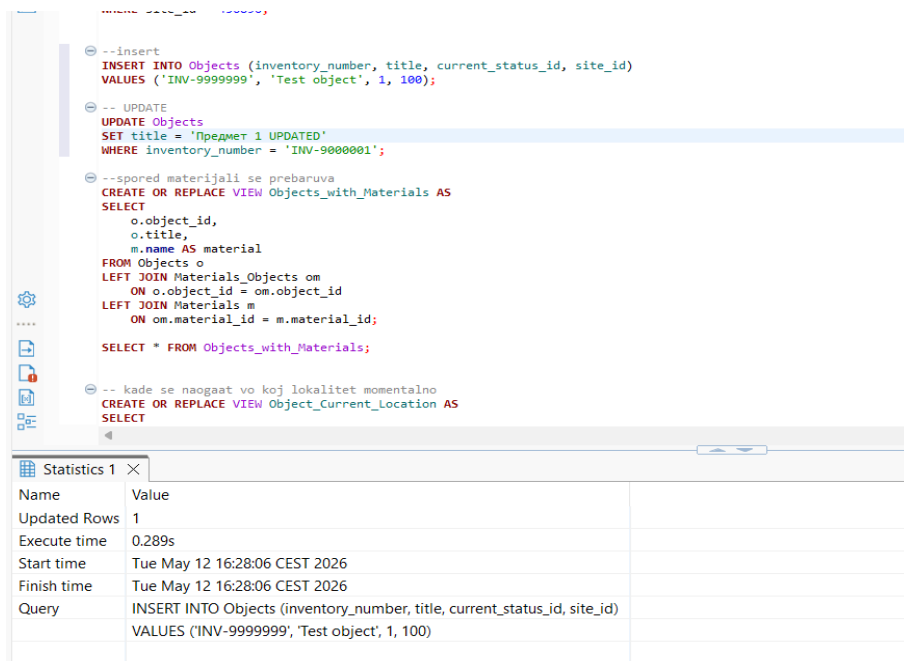
За оптимизација на перформансите, особено при филтрирање и пребарување, се користат следните индекси:

```
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



The screenshot displays a database management interface with a SQL editor and a statistics window. The SQL editor contains the following queries:

```
--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-9800001';

--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
```

The statistics window, titled "Statistics 1", shows the following data:

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)

<pre>-- UPDATE UPDATE Objects SET title = 'UPDATED TEST OBJECT' WHERE inventory_number = 'INV-9999999';</pre>													
<pre>--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;</pre>													
<pre>-- kade se naogaat vo koj lokalitet momentalno CREATE OR REPLACE VIEW Object_Current_Location AS SELECT</pre>													
<table border="1"> <thead> <tr> <th>Name</th><th>Value</th></tr> </thead> <tbody> <tr> <td>Updated Rows</td><td>1</td></tr> <tr> <td>Execute time</td><td>0.062s</td></tr> <tr> <td>Start time</td><td>Tue May 12 16:33:37 CEST 2026</td></tr> <tr> <td>Finish time</td><td>Tue May 12 16:33:37 CEST 2026</td></tr> <tr> <td>Query</td><td>UPDATE Objects SET title = 'UPDATED TEST OBJECT' WHERE inventory_number = 'INV-9999999'</td></tr> </tbody> </table>		Name	Value	Updated Rows	1	Execute time	0.062s	Start time	Tue May 12 16:33:37 CEST 2026	Finish time	Tue May 12 16:33:37 CEST 2026	Query	UPDATE Objects SET title = 'UPDATED TEST OBJECT' WHERE inventory_number = 'INV-9999999'
Name	Value												
Updated Rows	1												
Execute time	0.062s												
Start time	Tue May 12 16:33:37 CEST 2026												
Finish time	Tue May 12 16:33:37 CEST 2026												
Query	UPDATE Objects SET title = 'UPDATED TEST OBJECT' WHERE inventory_number = 'INV-9999999'												
	CEST en Writable Smart Insert 1336 : 1 [90] Sel: 9												

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.

```

-- bade se naogaaat vo koj lokalitet momentalno
CREATE OR REPLACE VIEW Object_Current_Location AS
SELECT
  o.object_id,
  o.title,
  rand_inst.name AS institution
FROM Objects o
CROSS JOIN LATERAL (
  SELECT i.name
  FROM Institutions i
  WHERE o.object_id = i.object_id
  ORDER BY random()
  LIMIT 1
) rand_inst;

EXPLAIN ANALYZE SELECT * FROM Object_Current_Location WHERE object_id = 1340;
-- bade baktuvaa referenc

```

Results 1

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

QUERY PLAN

1	Nested Loop (cost=13.93..21.97 rows=1 width=250) (actual time=0.031..0.032 rows=0 loops=1)
2	-> Index Scan using objects_pkey on objects o (cost=0.43..8.45 rows=1 width=32) (actual time=0.000..0.001 rows=1 loops=1)
3	Index Cond: (object_id = 1340)
4	-> Limit (cost=13.50..13.50 rows=1 width=226) (never executed)
5	-> Sort (cost=13.50..14.00 rows=200 width=226) (never executed)
6	Sort Key: (random())
7	-> Result (cost=0.00..12.50 rows=200 width=226) (never executed)
8	One-Time Filter: (o.object_id = o.object_id)
9	-> Seq Scan on institutions i (cost=0.00..12.00 rows=200 width=218) (never executed)
10	Planning Time: 0.268 ms
11	Execution Time: 0.073 ms

Value X

Text

Nested Loop (cost=13.93..21.97)

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 Institutions 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

```

VALUES (00000000, 0, CURRENT_DATE);

-- 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

```

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

```

Grid	Text	Record
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 loops=1)	
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	

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.

The screenshot shows a database IDE with a SQL script editor and a results pane. The SQL script includes a JOIN, a CREATE INDEX statement, and an EXPLAIN ANALYZE query. The results pane displays the execution plan for the query.

```
JOIN Culture c ON c.culture_id = oc.culture_id
JOIN Categories cat ON cat.category_id = oc.category_id;

CREATE INDEX idx_oc_object_id ON Object_Classification(object_id);

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

The execution plan shows a Nested Loop query with the following steps:

1. Nested Loop (cost=1.15..33.35 rows=1 width=468) (actual time=0.081..0.083 rows=0 loops=1)
2. -> Nested Loop (cost=1.00..25.12 rows=1 width=258) (actual time=0.081..0.082 rows=0 loops=1)
3. -> Nested Loop (cost=0.85..16.90 rows=1 width=48) (actual time=0.081..0.082 rows=0 loops=1)
4. -> Index Scan using objects_pkey on objects o (cost=0.43..8.45 rows=1 width=32) (actual time=0.080..0.080 rows=0 loops=1)
5. Index Cond: (object_id = 150)
6. -> Index Scan using idx_oc_object_id on object_classification oc (cost=0.43..8.45 rows=1 width=24) (never executed)
7. Index Cond: (object_id = 150)
8. -> Index Scan using culture_pkey on culture c (cost=0.15..8.17 rows=1 width=226) (never executed)
9. Index Cond: (culture_id = oc.culture_id)
10. -> Index Scan using categories_pkey on categories cat (cost=0.15..8.17 rows=1 width=226) (never executed)
11. Index Cond: (category_id = oc.category_id)
12. Planning Time: 272.882 ms
13. Execution Time: 0.142 ms

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

```

JOIN Categories cat ON cat.category_id = oc.category_id;

CREATE INDEX idx_oc_object_id ON Object_Classification(object_id);

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 = 1000;

--koi predmeti na koi izlozbi

```

Statistics 1	
Name	Value
Updated Rows	1
Execute time	0.206s
Start time	Tue May 12 22:00:45 CEST 2026
Finish time	Tue May 12 22:00:45 CEST 2026
Query	INSERT INTO Object_Classification (object_id, category_id, culture_id) VALUES (1649219, 1, 1)

<pre> 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 </pre>	
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

<pre> ORDER BY random() LIMIT 1) rand_id; EXPLAIN ANALYZE SELECT * FROM Exhibition_Objects LIMIT 10; --koj naucnik ili istrazuvas pobara da istrazuva predmet dali mu e odobreno momentalno </pre>	
Results 1	Value
Grid	Text
<p>QUERY PLAN</p> <p>1 Limit (cost=166.00..1826.81 rows=10 width=293) (actual time=1373.943..1392.322 rows=10 loops=1)</p> <p>2 -> Nested Loop (cost=166.00..331896328.30 rows=1998405 width=293) (actual time=1373.941..1392.316 rows=10 loops=1)</p> <p>3 -> Nested Loop (cost=152.50..304867900.67 rows=1998405 width=83) (actual time=1165.391..1183.509 rows=10 loops=1)</p> <p>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)</p> <p>5 -> Limit (cost=152.50..152.50 rows=1 width=59) (actual time=62.235..62.235 rows=1 loops=10)</p> <p>6 -> Sort (cost=152.50..165.00 rows=5000 width=59) (actual time=62.233..62.233 rows=1 loops=10)</p> <p>7 Sort Key: (random())</p> <p>8 Sort Method: top-N heapsort Memory: 25kB</p> <p>9 -> Result (cost=0.00..127.50 rows=5000 width=59) (actual time=10.071..61.315 rows=5000 loops=10)</p> <p>10 One-Time Filter: (o.object_id = o.object_id)</p> <p>11 -> Seq Scan on exhibitions e (cost=0.00..115.00 rows=5000 width=51) (actual time=10.062..60.573 rows=5000 loops=10)</p> <p>12 -> Limit (cost=13.50..13.50 rows=1 width=226) (actual time=20.878..20.878 rows=1 loops=10)</p> <p>13 -> Sort (cost=13.50..14.00 rows=200 width=226) (actual time=20.876..20.876 rows=1 loops=10)</p> <p>14 Sort Key: (random())</p> <p>15 Sort Method: top-N heapsort Memory: 25kB</p> <p>16 -> Result (cost=0.00..12.50 rows=200 width=226) (actual time=20.856..20.860 rows=21 loops=10)</p> <p>17 One-Time Filter: (o.object_id = o.object_id)</p> <p>18 -> Seq Scan on institutions i (cost=0.00..12.00 rows=200 width=218) (actual time=20.854..20.856 rows=21 loops=10)</p> <p>19 Planning Time: 675.957 ms</p> <p>20 Execution Time: 1485.615 ms</p>	
<p>Limit (cost=166.00..1826.81 rows=10 width=293) (actual time=1373.943..1392.322 rows=10 loops=1)</p>	

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);
```

EXPLAIN ANALYZE SELECT * FROM Exhibition_Objects WHERE exhibition_id = 204050;
EXPLAIN ANALYZE SELECT * FROM Exhibition_Objects WHERE exhibition_id = 4;

--koj naucnik ili istrazuvas pobaral da istrazuva predmet dali mu e odobreno momentalno
CREATE OR REPLACE VIEW Research_Access_Details AS
SELECT
o.full_name,
o.title AS object,
i.name AS institution,
i.status_name AS access_status,
(CONVERT_DATE + (random() * 400 - 200)::int) AS access_date
FROM generate_series(1, 25) g(id)
LEFT JOIN LATERAL (
SELECT full_name
FROM Users

Results 1 Results 2

EXPLAIN ANALYZE SELECT * FROM Exhibition_Objects WHERE exhibition_id = 204050;
EXPLAIN ANALYZE SELECT * FROM Exhibition_Objects WHERE exhibition_id = 4;

SQL QUERY PLAN

1	Nested Loop (cost=6.04..1200.44 rows=99 width=309) (actual time=500.889..27304.622 rows=100 loops=1)
2	-> Nested Loop (cost=5.62..364.39 rows=99 width=285) (actual time=0.210..242.477 rows=100 loops=1)
3	-> Nested Loop (cost=0.43..16.54 rows=1 width=277) (actual time=0.119..0.125 rows=1 loops=1)
4	-> Index Scan using exhibitions_pkey on exhibitions e (cost=0.28..8.30 rows=1 width=67) (actual time=0.067..0.071 rows=1 loops=1)
5	Index Cond: (exhibition_id = 4)
6	-> Index Scan using institutions_pkey on institutions i (cost=0.14..8.16 rows=1 width=226) (actual time=0.045..0.045 rows=1 loops=1)
7	Index Cond: (institution_id = e.location_institution_id)
8	-> Bitmap Heap Scan on object_exhibition oe (cost=5.19..346.86 rows=99 width=16) (actual time=0.088..242.156 rows=99 loops=1)
9	Recheck Cond: (exhibition_id = 4)
10	Heap Blocks: exact=98
11	-> Bitmap Index Scan on idx_oe_exhibition_id (cost=0.00..5.17 rows=99 width=0) (actual time=0.053..0.054 rows=99 loops=1)
12	Index Cond: (exhibition_id = 4)
13	-> Index Scan using objects_pkey on objects o (cost=0.43..8.45 rows=1 width=32) (actual time=270.611..270.611 rows=1 loops=1)
14	Index Cond: (object_id = oe.object_id)
15	Planning Time: 0.724 ms
16	Execution Time: 27304.836 ms

Activate Windows

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.

```

EXPLAIN ANALYZE SELECT * FROM Exhibition_Objects WHERE exhibition_id = 204050;
EXPLAIN ANALYZE SELECT * FROM Exhibition_Objects WHERE exhibition_id = 4;

--koj naucnik ili istrazuvas pobaral da istrazuva predmet dali mu e odobreno momentalno
CREATE OR REPLACE VIEW Research_Access_Details AS
SELECT
    u.full_name,
    o.title AS object,
    i.name AS institution,
    s.status_name AS access_status,
    (CURRENT_DATE + (random() * 400 - 200)::int) AS access_date
FROM generate_series(1, 25) s(id)

```

Results 1 Results 2 X

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

AZ QUERY PLAN

1	Nested Loop (cost=6.04..1200.44 rows=99 width=309) (actual time=0.016..0.018 rows=0 loops=1)
2	-> Nested Loop (cost=5.62..364.39 rows=99 width=285) (actual time=0.016..0.017 rows=0 loops=1)
3	-> Nested Loop (cost=0.43..16.54 rows=1 width=277) (actual time=0.016..0.017 rows=0 loops=1)
4	-> Index Scan using exhibitions_pkey on exhibitions e (cost=0.28..8.30 rows=1 width=67) (actual time=0.015..0.015 rows=0 loops=1)
5	Index Cond: (exhibition_id = 204050)
6	-> Index Scan using institutions_pkey on institutions i (cost=0.14..8.16 rows=1 width=226) (never executed)
7	Index Cond: (institution_id = e.location_institution_id)
8	-> Bitmap Heap Scan on object_exhibition oe (cost=5.19..346.86 rows=99 width=16) (never executed)
9	Recheck Cond: (exhibition_id = 204050)
10	-> Bitmap Index Scan on idx_oe_exhibition_id (cost=0.00..5.17 rows=99 width=0) (never executed)
11	Index Cond: (exhibition_id = 204050)
12	-> Index Scan using objects_pkey on objects o (cost=0.43..8.45 rows=1 width=32) (never executed)
13	Index Cond: (object_id = oe.object_id)
14	Planning Time: 0.891 ms
15	Execution Time: 0.079 ms

Refresh Save Cancel Export data 200 15 1

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

```

EXPLAIN ANALYZE SELECT * FROM Exhibition_Objects LIMIT 10;
EXPLAIN ANALYZE SELECT * FROM Exhibition_Objects WHERE exhibition_id = 204050;
EXPLAIN ANALYZE SELECT * FROM Exhibition_Objects WHERE exhibition_id = 4;

-- INSERT
INSERT INTO Object_Exhibition (object_id, exhibition_id)
VALUES (
    (SELECT MAX(object_id) FROM Objects),
    (SELECT MAX(exhibition_id) FROM Exhibitions)
);

-- UPDATE
UPDATE Object_Exhibition
SET exhibition_id = 2
WHERE object_id = 1439793;

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

```

Statistics 1 X

Name	Value
Updated Rows	1
Execute time	0.343s
Start time	Tue May 12 22:07:34 CEST 2026
Finish time	Tue May 12 22:07:35 CEST 2026
Query	INSERT INTO Object_Exhibition (object_id, exhibition_id) VALUES ((SELECT MAX(object_id) FROM Objects), (SELECT MAX(exhibition_id) FROM Exhibitions))

The screenshot shows a database management tool interface. The top pane displays SQL queries. The bottom pane shows a 'Statistics 1' window with the following data:

Name	Value
Updated Rows	1
Execute time	0.015s
Start time	Tue May 12 22:08:36 CEST 2026
Finish time	Tue May 12 22:08:36 CEST 2026
Query	UPDATE Object_Exhibition SET exhibition_id = 3 WHERE object_id = (SELECT MAX(object_id) FROM Object_Exhibition) AND exhibition_id = (SELECT MAX(exhibition_id) FROM Object_Exhibition);

At the bottom of the interface, there is a status bar with the following information: CEST | en | Writable | Smart Insert | 1445 : 1 [202] | Sel: 202 | 8 | ...

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;
```

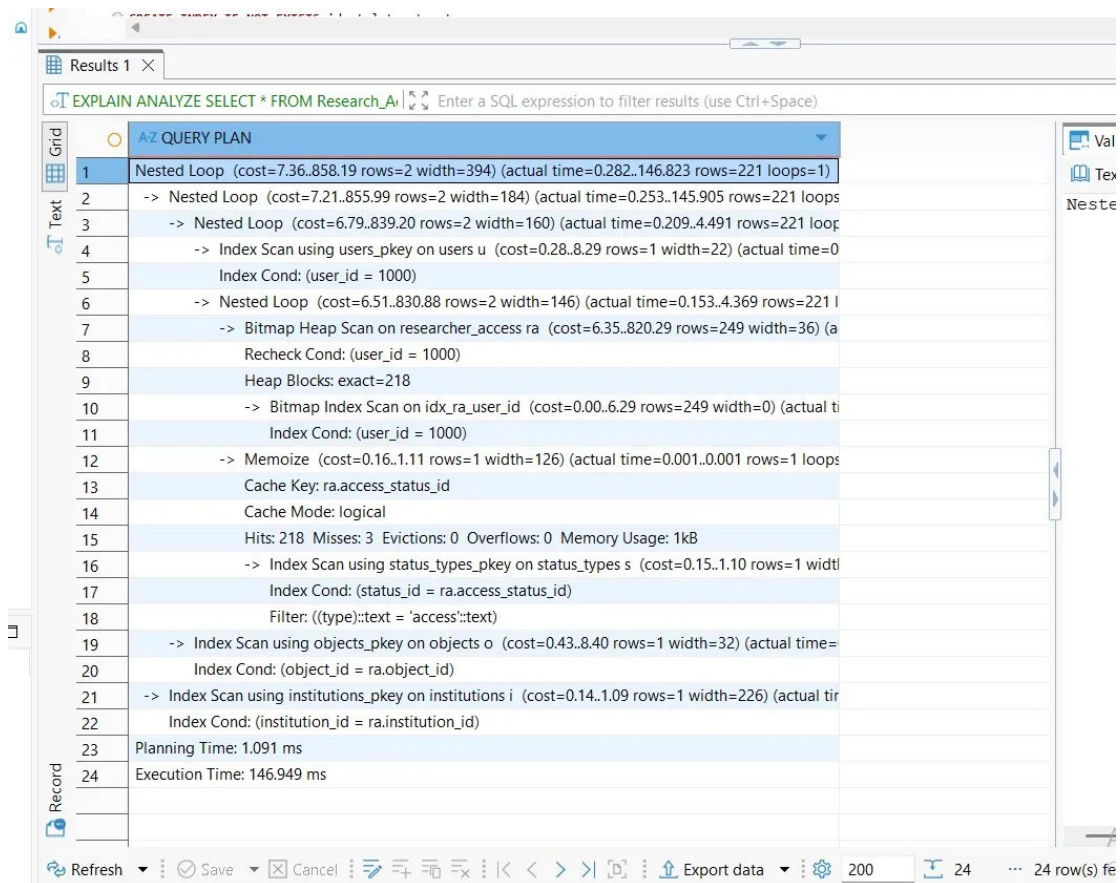
По оптимизацијата најдоброто постигнато време изнесува 146.949 ms.

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.



Grid	Text	Record
1	Nested Loop (cost=7.36..858.19 rows=2 width=394) (actual time=0.282..146.823 rows=221 loops=1)	
2	-> Nested Loop (cost=7.21..855.99 rows=2 width=184) (actual time=0.253..145.905 rows=221 loops=1)	
3	-> Nested Loop (cost=6.79..839.20 rows=2 width=160) (actual time=0.209..4.491 rows=221 loops=1)	
4	-> Index Scan using users_pkey on users u (cost=0.28..8.29 rows=1 width=22) (actual time=0.000..0.001 rows=1 loops=1)	
5	Index Cond: (user_id = 1000)	
6	-> Nested Loop (cost=6.51..830.88 rows=2 width=146) (actual time=0.153..4.369 rows=221 loops=1)	
7	-> Bitmap Heap Scan on researcher_access ra (cost=6.35..820.29 rows=249 width=36) (actual time=0.000..0.001 rows=221 loops=1)	
8	Recheck Cond: (user_id = 1000)	
9	Heap Blocks: exact=218	
10	-> Bitmap Index Scan on idx_ra_user_id (cost=0.00..6.29 rows=249 width=0) (actual time=0.000..0.001 rows=221 loops=1)	
11	Index Cond: (user_id = 1000)	
12	-> Memoize (cost=0.16..1.11 rows=1 width=126) (actual time=0.001..0.001 rows=1 loops=1)	
13	Cache Key: ra.access_status_id	
14	Cache Mode: logical	
15	Hits: 218 Misses: 3 Evictions: 0 Overflows: 0 Memory Usage: 1kB	
16	-> Index Scan using status_types_pkey on status_types s (cost=0.15..1.10 rows=1 width=126) (actual time=0.000..0.001 rows=1 loops=1)	
17	Index Cond: (status_id = ra.access_status_id)	
18	Filter: ((type)::text = 'access'::text)	
19	-> Index Scan using objects_pkey on objects o (cost=0.43..8.40 rows=1 width=32) (actual time=0.000..0.001 rows=1 loops=1)	
20	Index Cond: (object_id = ra.object_id)	
21	-> Index Scan using institutions_pkey on institutions i (cost=0.14..1.09 rows=1 width=226) (actual time=0.000..0.001 rows=1 loops=1)	
22	Index Cond: (institution_id = ra.institution_id)	
23	Planning Time: 1.091 ms	
24	Execution Time: 146.949 ms	

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

```

-- Create index on Researcher_Access
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
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;

-- Create or replace view
CREATE OR REPLACE VIEW Treatment_History AS
SELECT
  t.object_id,
  o.title,
  t.treatment_date,
  tsl.step_number,
  tsl.step_description,
  ...

```

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, user_id, object_id, 1 FROM Users u JOIN Objects o ON o.object_id = u.user_id LIMIT 1;

Activate
Go to Setting

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

-- Insert into Researcher_Access
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;

```

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

The screenshot displays a database IDE interface. At the top, a SQL query is shown: `JOIN Users u ON u.user_id = tsl.performed_by_user; EXPLAIN ANALYZE SELECT * FROM Treatment_History WHERE object_id = 100;`. Below the query, a comment in Macedonian reads: `--koj avtor koi publikacii gi napravil`. Further down, a `CREATE MATERIALIZED VIEW` statement is visible: `CREATE MATERIALIZED VIEW Publications_with_Authors_MV AS SELECT p.title, a.full_name AS author FROM Publication_Authors pa JOIN Publications p ON p.publication_id = pa.publication_id JOIN Authors a ON a.author_id = pa.author_id; SELECT * FROM Publications_with_Authors_MV;`. The main panel shows the 'Results 1' tab with a query plan for the `EXPLAIN ANALYZE` query. The plan details a nested loop join with various index scans and their associated costs and execution times. The bottom status bar indicates '13 row(s) fetched'.

```
JOIN Users u ON u.user_id = tsl.performed_by_user;
EXPLAIN ANALYZE SELECT * FROM Treatment_History WHERE object_id = 100;

--koj avtor koi publikacii gi napravil
CREATE MATERIALIZED VIEW Publications_with_Authors_MV AS
SELECT
  p.title,
  a.full_name AS author
FROM Publication_Authors pa
JOIN Publications p
  ON p.publication_id = pa.publication_id
JOIN Authors a
  ON a.author_id = pa.author_id;
SELECT * FROM Publications_with_Authors_MV;
```

Results 1

EXPLAIN ANALYZE SELECT * FROM Treatment_History WHERE object_id = 100; Enter a SQL expression to filter results (use Ctrl+Space)

AZ QUERY PLAN

1	Nested Loop (cost=1.56..72.95 rows=5 width=75) (actual time=0.255..0.257 rows=0 loops=1)
2	-> Nested Loop (cost=1.29..41.47 rows=5 width=69) (actual time=0.254..0.256 rows=0 loops=1)
3	-> Nested Loop (cost=0.85..16.90 rows=1 width=44) (actual time=0.254..0.255 rows=0 loops=1)
4	-> Index Scan using idx_treatments_object on treatments t (cost=0.42..8.44 rows=1 width=20) (actual time=0.254..0.255 rows=0 loops=1)
5	Index Cond: (object_id = 100)
6	-> Index Scan using objects_pkey on objects o (cost=0.43..8.45 rows=1 width=32) (never executed)
7	Index Cond: (object_id = 100)
8	-> Index Scan using idx_tsl_treatment on treatment_step_log tsl (cost=0.43..24.52 rows=5 width=41) (never executed)
9	Index Cond: (treatment_id = t.treatment_id)
10	-> Index Scan using users_pkey on users u (cost=0.28..6.30 rows=1 width=22) (never executed)
11	Index Cond: (user_id = tsl.performed_by_user)
12	Planning Time: 1.367 ms
13	Execution Time: 0.549 ms

Refresh Save Cancel Export data 200 13 13 row(s) fetched

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

The screenshot shows a database IDE with a SQL editor and a statistics window. The SQL editor contains the following queries:

```
SELECT
    t.object_id,
    o.title,
    t.treatment_date,
    tsl.step_number,
    tsl.step_description,
    u.full_name
FROM Treatments t
JOIN Objects o ON o.object_id = t.object_id
JOIN Treatment_Step_Log tsl ON tsl.treatment_id = t.treatment_id
JOIN Users u ON u.user_id = tsl.performed_by_user;

EXPLAIN ANALYZE SELECT * FROM Treatment_History WHERE object_id = 100;

-- INSERT
INSERT INTO Treatments (object_id, treatment_date, description)
SELECT object_id, CURRENT_DATE, 'Test treatment'
FROM Objects
WHERE object_id BETWEEN 1 AND 10
LIMIT 1;

-- UPDATE
UPDATE Treatments
SET description = 'Updated treatment'
WHERE object_id = 1000;

-- koj avtor koi publikacii gi napravil
```

The statistics window, titled "Statistics 1", shows the following data:

Name	Value
Updated Rows	0
Execute time	0.016s
Start time	Tue May 12 22:23:18 CEST 2026
Finish time	Tue May 12 22:23:18 CEST 2026
Query	INSERT INTO Treatments (object_id, treatment_date, description) SELECT object_id, CURRENT_DATE, 'Test treatment' FROM Objects WHERE object_id BETWEEN 1 AND 10 LIMIT 1

```

INSERT INTO Treatments (object_id, treatment_date, description)
SELECT object_id, CURRENT_DATE, 'Test treatment'
FROM Objects
WHERE object_id BETWEEN 1 AND 10
LIMIT 1;

-- UPDATE
UPDATE Treatments
SET description = 'Update treatment',
    treatment_date = CURRENT_DATE
WHERE object_id = (
    SELECT object_id
    FROM Objects
    WHERE object_id BETWEEN 1 AND 10
    LIMIT 1
)
AND description = 'Test treatment';

--koj avtor koi publikacii gi napravil
CREATE OR REPLACE VIEW Publications_with_Authors AS
SELECT

```

Name	Value
Updated Rows	0
Execute time	0.011s
Start time	Tue May 12 22:30:03 CEST 2026
Finish time	Tue May 12 22:30:03 CEST 2026
Query	UPDATE Treatments SET description = 'Update treatment', treatment_date = CURRENT_DATE WHERE object_id = (SELECT object_id FROM Objects WHERE object_id BETWEEN 1 AND 10 LIMIT 1 AND description = 'Test treatment';

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.

```

EXPLAIN ANALYZE SELECT * FROM Treatment_History WHERE object_id = 100;

--koj avtor koi publikacii gi napravil
DROP MATERIALIZED VIEW IF EXISTS Publications_with_Authors_MV CASCADE;

CREATE OR REPLACE VIEW Publications_with_Authors AS
SELECT
    pa.publication_id,
    p.title,
    a.full_name AS author
FROM Publication_Authors pa
JOIN Publications p ON p.publication_id = pa.publication_id
JOIN Authors a ON a.author_id = pa.author_id;

EXPLAIN ANALYZE SELECT * FROM Publications_with_Authors WHERE publication_id = 100;

```

Step	Operation	Cost	Rows	Width	Actual Time	Actual Rows	Actual Loops
1	Nested Loop	1.14	21.20	rows=1 width=112	1158.195..1158.204	rows=1	loops=1
2	-> Nested Loop	0.84	12.89	rows=1 width=94	1009.399..1009.407	rows=1	loops=1
3	-> Index Only Scan using publication_authors_pkey on publication_authors pa	0.42	4.44	rows=1 width=16			
4	Index Cond: (publication_id = 100)						
5	Heap Fetches: 0						
6	-> Index Scan using publications_pkey on publications p	0.42	8.44	rows=1 width=86	498.701..498.707	rows=1	loops=1
7	Index Cond: (publication_id = 100)						
8	-> Index Scan using authors_pkey on authors a	0.29	8.31	rows=1 width=34	148.785..148.785	rows=1	loops=1
9	Index Cond: (author_id = pa.author_id)						
10	Planning Time	1216.740	ms				
11	Execution Time	1158.266	ms				

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

The screenshot displays a database IDE with the following SQL queries in the editor:

```
EXPLAIN ANALYZE SELECT * FROM Treatment_History WHERE object_id = 100;  
--koj avtor koi publikacii gi napravil  
CREATE OR REPLACE VIEW Publications_with_Authors AS  
SELECT  
  pa.publication_id,  
  p.title,  
  a.full_name AS author  
FROM Publication_Authors pa  
JOIN Publications p ON p.publication_id = pa.publication_id  
JOIN Authors a ON a.author_id = pa.author_id;  
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;  
EXPLAIN ANALYZE SELECT * FROM Publications_with_Authors WHERE publication_id = 100;
```

The results pane shows the execution plan for the query `EXPLAIN ANALYZE SELECT * FROM Publications`:

Step	Operation	Cost	Rows	Width	Actual Time	Actual Rows	Loops
1	Nested Loop	1.14..21.20	1	112	24.419..24.430	1	1
2	-> Nested Loop	0.84..12.89	1	94	0.212..0.221	1	1
3	-> Index Only Scan using publication_authors_pkey on publication_authors pa	0.42..4.44	1	16			
4	Index Cond: (publication_id = 100)						
5	Heap Fetches: 0						
6	-> Index Scan using publications_pkey on publications p	0.42..8.44	1	86	0.061..0.065		
7	Index Cond: (publication_id = 100)						
8	-> Index Scan using authors_pkey on authors a	0.29..8.31	1	34	24.201..24.202	1	1
9	Index Cond: (author_id = pa.author_id)						
10	Planning Time: 2.412 ms						
11	Execution Time: 24.479 ms						

The status bar at the bottom indicates 11 row(s) fetched on 2026-05-06 at 12:14:39:49440, with 0 inserts and 0 updates.

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

The top screenshot shows a SQL query window with the following code:

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

EXPLAIN ANALYZE SELECT * FROM Publications_with_Authors WHERE publication_id = 100;

-- INSERT
INSERT INTO Publication_Authors (publication_id, author_id)
VALUES (2, 50000);
```

The bottom screenshot shows the same query window with the following code:

```
ANALYZE Publication_Authors;
ANALYZE Publications;

EXPLAIN ANALYZE SELECT * FROM Publications_with_Authors WHERE publication_id = 100;

-- INSERT
INSERT INTO Publication_Authors (publication_id, author_id)
VALUES (2, 50000);

UPDATE Publication_Authors
SET author_id = 49999
WHERE publication_id = 2
AND author_id = 50000;
```

Below the code, a 'Statistics 1' window displays the execution details for the highlighted query:

Name	Value
Updated Rows	1
Execute time	1.651s
Start time	Tue May 12 23:07:55 CEST 2026
Finish time	Tue May 12 23:07:56 CEST 2026
Query	INSERT INTO Publication_Authors (publication_id, author_id) VALUES (2, 50000)

The bottom screenshot shows a similar 'Statistics 1' window for the UPDATE query:

Name	Value
Updated Rows	0
Execute time	0.02s
Start time	Tue May 12 23:09:23 CEST 2026
Finish time	Tue May 12 23:09:23 CEST 2026
Query	UPDATE Publication_Authors SET author_id = 49999 WHERE publication_id = 2 AND author_id = 50000

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.

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);

Results 1 X

EXPLAIN ANALYZE SELECT * FROM Heritage_Full_Overview WHERE site_id = 100 LIMIT 10; Enter a SQL expression to filter results (use Ctrl+Space)

QUERY PLAN

1	Limit (cost=1.01..1653.29 rows=10 width=347) (actual time=340.394..4055.268 rows=10 loops=1)
2	-> Nested Loop (cost=1.01..288158.78 rows=1744 width=347) (actual time=340.392..4055.255 rows=10 loops=1)
3	-> Nested Loop (cost=0.58..274274.70 rows=1744 width=331) (actual time=340.335..3096.154 rows=10 loops=1)
4	-> Seq Scan on fragments f (cost=0.00..274228.20 rows=1744 width=56) (actual time=340.164..3095.916 rows=10 loops=1)
5	Filter: (site_id = 100)
6	Rows Removed by Filter: 47805
7	-> Materialize (cost=0.58..24.70 rows=1 width=283) (actual time=0.018..0.019 rows=1 loops=1)
8	-> Nested Loop (cost=0.58..24.70 rows=1 width=283) (actual time=0.159..0.165 rows=1 loops=1)
9	-> Nested Loop (cost=0.43..16.50 rows=1 width=173) (actual time=0.120..0.124 rows=1 loops=1)
10	-> Index Scan using sites_pkey on sites s (cost=0.28..8.30 rows=1 width=63) (actual time=0.073..0.075 rows=1 loops=1)
11	Index Cond: (site_id = 100)
12	-> Index Scan using regions_pkey on regions r (cost=0.15..8.17 rows=1 width=126) (actual time=0.038..0.040 rows=1 loops=1)
13	Index Cond: (region_id = s.region_id)
14	-> Index Scan using protection_status_pkey on protection_status ps (cost=0.15..8.17 rows=1 width=126) (actual time=0.038..0.040 rows=1 loops=1)
15	Index Cond: (protection_status_id = s.protection_status_id)
16	-> Index Scan using objects_pkey on objects o (cost=0.43..7.96 rows=1 width=32) (actual time=95.901..95.901 rows=1 loops=1)
17	Index Cond: (object_id = f.object_id)
18	Planning Time: 1.215 ms
19	Execution Time: 4055.439 ms

Value X

Text

Limit (cost=1.01..1653.29 rows=10 width=347)

Refresh Save Cancel

Export data

200

19

19 row(s) fetched: 4.112s; on 2026-05-06

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)

A-Z 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=0.000..0.000 rows=10 loops=1)
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 rows=1 loops=1)
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..0.062 rows=1 loops=1)
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) (actual time=0.056..0.056 rows=1 loops=1)
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 loops=1)
16	Index Cond: (object_id = f.object_id)
17	Planning Time: 223.988 ms
18	Execution Time: 1.264 ms

Record

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

ce

CREATE INDEX idx_pa_author_id ON Publication_Authors(author_id);

JOIN Protection_Status ps ON s.protection_status_id = ps.protection_status_id;

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

-- INSERT

INSERT INTO Fragments (
description,
site_id,
object_id,
status_id,
found_by_user_id,
discovery_date
)
SELECT
'Нов фрагмент',
1,
object_id,
1,
1,
CURRENT_DATE
FROM Objects
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

Statistics 1 X

Name	Value
Updated Rows	1
Execute time	2.152s
Start time	Tue May 12 23:14:17 CEST 2026
Finish time	Tue May 12 23:14:19 CEST 2026
Query	INSERT INTO Fragments (description, site_id, object_id, status_id

CESTenWritableSmart Insert1238 : 10 : 434

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
fragment_id

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 FROM Fragments

CEST

en

Writable

Smart Insert

1258 : 1 [139]

Sel: 139