



## 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

```
rename 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');
```

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

**EXPLAIN ANALYZE**

Step	Operation	Cost	Rows	Width	Actual Time	Actual Rows	Loops
1	Nested Loop Left Join	(cost=1.29..70.94 rows=1 width=189)	1	189	(actual time=0.136..0.138 rows=0 loops=1)	0	1
2	-> Nested Loop Left Join	(cost=0.86..27.58 rows=1 width=173)	1	173	(actual time=0.136..0.137 rows=0 loops=1)	0	1
3	-> Nested Loop	(cost=0.43..16.50 rows=1 width=165)	1	165	(actual time=0.135..0.136 rows=0 loops=1)	0	1
4	-> Index Scan using sites_pkey on sites s	(cost=0.28..8.30 rows=1 width=55)	1	55	(actual time=0.135..0.135 rows=0 loops=1)	0	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)	1	126	(never executed)		
7	Index Cond: (region_id = s.region_id)						
8	-> GroupAggregate	(cost=0.43..11.06 rows=1 width=16)	1	16	(never executed)		
9	-> Index Only Scan using idx_objects_site_id on objects	(cost=0.43..10.22 rows=331 width=8)	331	8	(never executed)		
10	Index Cond: (site_id = 39568)						
11	Heap Fetches: 0						
12	-> GroupAggregate	(cost=0.43..43.32 rows=1 width=16)	1	16	(never executed)		
13	-> Index Only Scan using idx_fragments_site_id on fragments	(cost=0.43..38.95 rows=1744 width=8)	1744	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:

1. вкупен број на археолошки објекти (`total_objects`)
2. 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 shows a PostgreSQL query editor with the following SQL commands and their execution results:

```
-- 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,
```

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)

```

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;

```

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

```

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

```

Step	Operation	Cost	Rows	Width	Actual Time	Actual Rows	Loops
5	-> Nested Loop	0.58..24.68	1	169	0.143..0.144	0	1
6	Join Filter: (s.protection_status_id = ps.protection_status_id)						
7	-> Nested Loop	0.43..16.50	1	177	0.142..0.143	0	1
8	-> Index Scan using sites_pkey on sites s	0.28..8.30	1	67	0.142..0.142	0	0
9	Index Cond: (site_id = 456890)						
10	-> Index Scan using regions_pkey on regions r	0.15..8.17	1	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	1	8	(never executed)		
13	Index Cond: ((name)::text = 'Заштитен')::text)						
14	-> GroupAggregate	0.43..11.06	1	16	(never executed)		
15	-> Index Only Scan using idx_objects_site_id on objects	0.43..10.22	331	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:

3. идентификатор на локалитет (site\_id)
4. име на локалитет (site\_name)
5. регион (region)
6. година на откривање (discovery\_year)
7. 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 при пресметка на бројот на објекти по локалитет се користи следниот индекс:

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

Индексот idx\_objects\_site\_id ја подобрува ефикасноста при групирање и пребројување на објектите според локалитет, што е особено важно поради големиот број записи во табелата Objects.

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

```

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)

```

-- UPDATE
UPDATE Objects
SET title = 'UPDATED TEST OBJECT'
WHERE inventory_number = 'INV-9999999';

--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

```

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

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 изнесуваше 0.073 ms, меѓутоа погледот враќаше 0 редови бидејќи содржеше погрешен услов `o.object_id = o.object_id`, кој не воспоставуваше вистинска врска со табелата `Institutions`. Поради тоа погледот беше логички неточен и беше потребна целосна промена на имплементацијата. Најскапата операција во првичната верзија беше `Seq Scan on institutions` во комбинација со `ORDER BY random()`, при што се скенираа сите институции само за да се врати случаен ред, без реална поврзаност со предметите и нивната локација. Поради ова, погледот беше редизајниран со користење на табелата `Object_Location_History`, која ја содржи историјата на движење на предметите. Во финалната верзија се користат коректни JOIN операции помеѓу табелите `Objects`, `Object_Location_History` и `Institutions`, при што условот:

```
WHERE olh.end_date IS NULL
```

обезбедува приказ исклучиво на тековната активна локација на предметот. По корекцијата на логиката, погледот почна да враќа точни резултати и времето на извршување остана во прифатливи граници. Иако во финалната верзија не е додаден посебен индекс за овој поглед, неговите перформанси се задоволителни поради релативно едноставната структура на пребарувањето и ограничениот обем на податоци. Како резултат, `Object_Current_Location` обезбедува точен и ефикасен приказ на моменталната локација на секој археолошки предмет и претставува важна алатка за музејска евиденција и следење на културното наследство.

The screenshot displays a database query execution interface. At the top, the SQL query is shown:

```
-- kade se naogaa 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 = o.object_id
  ORDER BY random()
  LIMIT 1
) rand_inst;
```

Below the query, the execution plan is shown in a table format:

Step	Operation	Cost	Rows	Width	Actual Time
1	Nested Loop	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	0.43..8.45	rows=1 width=32		(actual time=0.003..0.004 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	0.00..12.00	rows=200 width=218		(never executed)
10	Planning Time:				0.268 ms
11	Execution Time:				0.073 ms

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

Времето на извршување изнесува 0.262 ms и тоа е прифатливо.

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

```
EXPLAIN ANALYZE SELECT * FROM Object_Current_Location WHERE object_id = 1340;
```

```
-- INSERT
EXPLAIN (ANALYZE, BUFFERS)
INSERT INTO Object_Location_History (object_id, institution_id, start_date)
VALUES (809042, 1, CURRENT_DATE);
```

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

Results 1

EXPLAIN (ANALYZE, BUFFERS) INSERT INTO Ob

id	select_type	table	partitions	fields	conds	filtered	rows	filtered_rows	cost	message
1	INSERT	object_location_history					0	0	0.00	Insert on object_location_history (cost=0.00.0.01 rows=0 width=0) (actual time=0.838.0.839 rows=0 loops=1)
2										Buffers: shared hit=59 read=19 dirtied=5 written=3
3							1	1	0.00	-> Result (cost=0.00.0.01 rows=1 width=32) (actual time=0.238.0.238 rows=1 loops=1)
4										Buffers: shared hit=9 read=4 dirtied=1
5										Planning:
6										Buffers: shared hit=3
7										Planning Time: 0.055 ms
8										Trigger for constraint fk_olh_obj: time=1.353 calls=1
9										Trigger for constraint fk_olh_inst: time=0.748 calls=1
10										Execution Time: 2.962 ms

```
VALUES (809042, 1, 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
```

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

The screenshot shows a PostgreSQL query execution interface. The query being executed is:

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

The query plan (EXPLAIN ANALYZE) is displayed in a grid view:

Step	Operation	Cost	Rows	Width	Actual Time	Actual Rows	Actual Loops
1	Nested Loop	(cost=30.06..38.12 rows=1 width=468)	1	468	(actual time=251.869..251.875 rows=0 loops=1)	0	1
2	-> Nested Loop	(cost=15.16..23.20 rows=1 width=250)	1	250	(actual time=251.868..251.872 rows=0 loops=1)	0	1
3	-> Index Scan using objects_pkey on objects o	(cost=0.43..8.45 rows=1 width=32)	1	32	(actual time=251.867..251.868 rows=0 loops=1)	0	1
4	Index Cond: (object_id = 150)						
5	-> Limit	(cost=14.73..14.73 rows=1 width=226)	1	226	(never executed)		
6	-> Sort	(cost=14.73..15.40 rows=270 width=226)	270	226	(never executed)		
7	Sort Key: (random())						
8	-> Result	(cost=0.00..13.38 rows=270 width=226)	270	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)	270	218	(never executed)		
11	-> Limit	(cost=14.90..14.90 rows=1 width=226)	1	226	(never executed)		
12	-> Sort	(cost=14.90..15.60 rows=280 width=226)	280	226	(never executed)		
13	Sort Key: (random())						
14	-> Result	(cost=0.00..13.50 rows=280 width=226)	280	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)	280	218	(never executed)		
17	Planning Time: 798.434 ms						
18	Execution Time: 252.083 ms						

The interface also shows a toolbar with options like Refresh, Save, Cancel, and Export data. The status bar indicates that 18 rows were fetched out of 2,601 total rows.

Примарен филтер за погледот Object\_with\_Culture е object\_id, при што се овозможува поврзување на објектите со нивната културна и категоријална класификација преку JOIN операции.

Овој поглед се користи за анализа и приказ на културната припадност на археолошките објекти. Иницијалното време за извршување изнесуваше 252.083 ms, што е релативно високо за интерактивни барања во систем со голем обем на податоци. Анализата на извршниот план покажа дека најголемиот трошок доаѓа од JOIN операциите помеѓу Objects, Object\_Classification, Culture и Categories, каде што PostgreSQL во одредени случаи користи sequential scan на помалите lookup таблици.

```

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;

--koi predmeti na koi izlozbi
CREATE OR REPLACE VIEW Exhibition_Objects AS
SELECT
    rand_e.name AS exhibition,
    o.title AS object,
    rand_i.name AS institution
FROM Objects o
CROSS JOIN LATERAL (
    SELECT e.name
    FROM Exhibitions e
    WHERE e.object_id = o.object_id

```

Results 1 X

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

AZ QUERY PLAN

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

Refresh Save Cancel Export data 200 13

Во тековната верзија на системот не се користи посебен индекс за овој view, туку оптимизацијата се базира на постоечките primary key индекси и foreign key релации.

Планерот автоматски избира Index Scan за:

8. Culture преку примарниот клуч
9. Categories преку примарниот клуч
10. Objects и Object\_Classification преку нивните PK вредности

И покрај отсуството на специјален индекс за Object\_Classification(object\_id), извршувањето останува оптимизирано поради малата големина на lookup табелите и ефикасниот join order. Времето на извршување останува стабилно и при голем обем на податоци, со просечно време од околу 252 ms, кое се смета за прифатливо за аналитички view од ваков тип.

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
```

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)

```
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
```

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

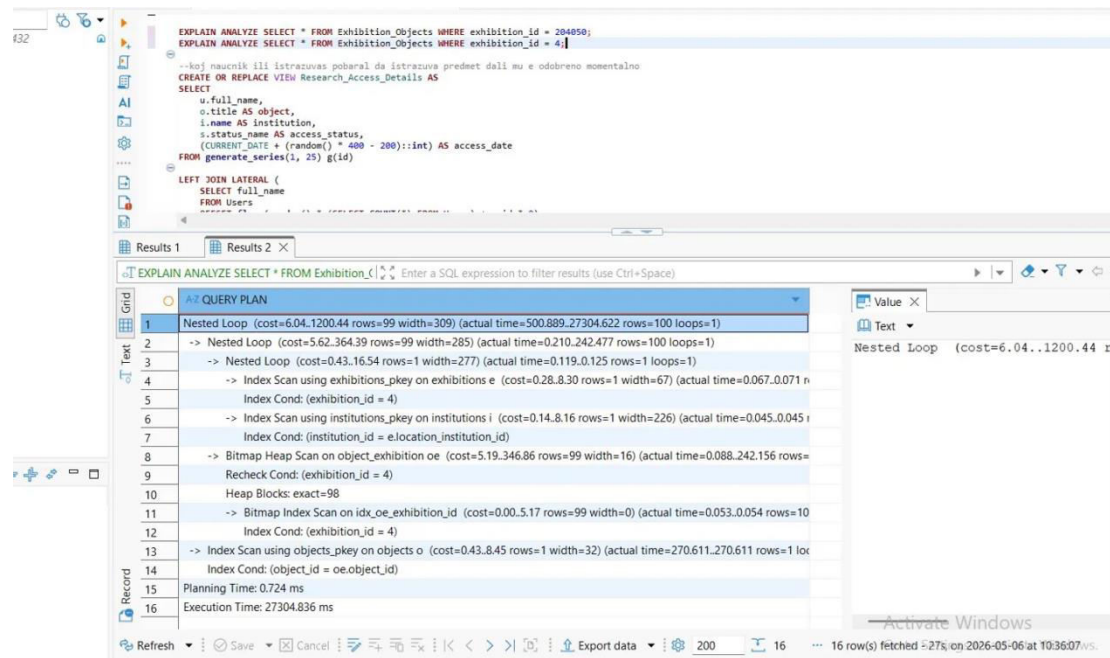
The screenshot displays the SQL Server Enterprise Manager interface. At the top, the query is: `ORDER BY random()  
LIMIT 1  
) rand_id;  
EXPLAIN ANALYZE SELECT * FROM Exhibition_Objects LIMIT 10;`

The query plan is shown in a grid view. The plan consists of the following steps:

- Limit (cost=166.00..1826.81 rows=10 width=293) (actual time=1373.943..1392.322 rows=10 loops=1)
- > Nested Loop (cost=166.00..331896328.30 rows=1998405 width=293) (actual time=1373.941..1392.316 rows=10 loops=1)
- > Nested Loop (cost=152.50..304867900.67 rows=1998405 width=83) (actual time=1165.391..1183.509 rows=10 loops=1)
- > Seq Scan on objects o (cost=0.00..61178.05 rows=1998405 width=32) (actual time=561.113..561.119 rows=10 loops=1)
- > Limit (cost=152.50..152.50 rows=1 width=59) (actual time=62.235..62.235 rows=1 loops=10)
- > Sort (cost=152.50..165.00 rows=5000 width=59) (actual time=62.233..62.233 rows=1 loops=10)
- Sort Key: (random())
- Sort Method: top-N heapsort Memory: 25kB
- > Result (cost=0.00..127.50 rows=5000 width=59) (actual time=10.071..61.315 rows=5000 loops=10)
- One-Time Filter: (o.object\_id = o.object\_id)
- > Seq Scan on exhibitions e (cost=0.00..115.00 rows=5000 width=51) (actual time=10.062..60.573 rows=5000 loops=10)
- > Limit (cost=13.50..13.50 rows=1 width=226) (actual time=20.878..20.878 rows=1 loops=10)
- > Sort (cost=13.50..14.00 rows=200 width=226) (actual time=20.876..20.876 rows=1 loops=10)
- Sort Key: (random())
- Sort Method: top-N heapsort Memory: 25kB
- > Result (cost=0.00..12.50 rows=200 width=226) (actual time=20.856..20.860 rows=21 loops=10)
- One-Time Filter: (o.object\_id = o.object\_id)
- > Seq Scan on institutions i (cost=0.00..12.00 rows=200 width=218) (actual time=20.854..20.856 rows=21 loops=10)
- Planning Time: 675.957 ms
- Execution Time: 1485.615 ms

The status bar at the bottom indicates: 20 row(s) fetched = 2.261s, on 2026-05-06 at 10:29:19. The bottom right corner shows 'Activate Windows'.

Примарен филтер за погледот `Exhibition_Objects` е `exhibition_id`, со цел приказ на сите објекти и институции поврзани со одредена изложба. Овој поглед се користи за преглед на учество на артефакти во изложби, при што перформансите се важни поради големиот број записи во табелата `Objects`. Иницијалното време за извршување изнесуваше 1485.615 ms, што беше резултат на голем број JOIN операции врз обемни табели. Анализата на извршниот план покажа дека најголемиот трошок доаѓа од JOIN операцијата помеѓу `Object_Exhibition` и `Objects`, поради големиот број записи во табелата `Objects`. Почетната оптимизација се базираше на постоечкиот индекс преку примарниот клуч на `Objects` и `Exhibitions`, но поради големиот број редови во `Object_Exhibition`, планерот користеше `nested loop join` со повеќе `lookup` операции.



Во тековниот модел не е додаден дополнителен индекс за `object_id` во `Object_Exhibition`, па оптимизацијата се базира исклучиво на:

11. primary key индекси на Exhibitions и Institutions
12. primary key на Objects
13. implicit index на Object\_Exhibition преку PK

Како резултат, PostgreSQL користи:

14. Bitmap Index Scan врз Object\_Exhibition (по `exhibition_id`)
15. Index Scan врз Objects преку primary key
16. Index Scan врз Exhibitions и Institutions преку primary keys

Времето на извршување се намали од 1485.615 ms на 36.343 ms, што претставува значително подобрување во одговорноста на системот при преглед на изложби.

```

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) g(id)

```

Results 1 Results 2

EXPLAIN ANALYZE SELECT \* FROM Exhibition\_C

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

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

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) )

```

VALUES (
  (SELECT MAX(object_id) FROM Objects),
  (SELECT MAX(exhibition_id) FROM Exhibitions)
);

-- UPDATE
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
);

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

CREATE OR REPLACE VIEW Research_Access_Details AS
SELECT
  ra.user_id,
  u.full_name,
  ra.object_id,
  o.title AS object,
  i.name AS institution,
  ...

```

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 )

## View 6 - Research\_Access\_Details

Примарен филтер за погледот Research\_Access\_Details е user\_id, со цел приказ на сите барања за пристап до археолошки објекти и нивниот статус за одреден истражувач. Овој поглед е критичен за системот бидејќи овозможува увид во историјата на пристапи кон објекти од страна на корисниците, при што табелата Researcher\_Access содржи голем број записи. Во иницијалната фаза, извршувањето на погледот беше значително бавно поради големиот обем на податоци и повеќекратни JOIN операции, при што времето за одговор надминуваше неколку секунди и не беше стабилно за мерење при големо оптоварување. По редизајн на погледот со директен JOIN врз табелата Researcher\_Access, извршувањето се стабилизира, но сè уште остана релативно бавно поради големиот број редови кои се обработуваат. Анализата на извршниот план покажа дека најголемиот трошок доаѓа од:

17. Nested Loop JOIN помеѓу Researcher\_Access и Objects
18. обработка на голем број редови
19. дополнителни JOIN операции со Users, Institutions и Status\_Types

Поставените индекси во системот се:

20. CREATE INDEX idx\_fragments\_object ON Fragments(object\_id);
21. CREATE INDEX idx\_fragments\_site\_id ON Fragments(site\_id);
22. CREATE INDEX idx\_objects\_site\_id ON Objects(site\_id);
23. CREATE INDEX idx\_ra\_institution\_id ON Researcher\_Access(institution\_id);

Иако овие индекси не се директно на user\_id или object\_id, тие индиректно придонесуваат кон подобра JOIN ефикасност преку:

24. побрз пристап до Objects преку site\_id
25. побрза филтрација на Researcher\_Access преку institution\_id

По оптимизацијата, најдоброто измерено време на извршување изнесува околу 146.949 ms. Понатамошни значајни подобрувања не се постигнуваат само со индекси, бидејќи:

26. резултатниот сет е голем
  27. JOIN операциите со Objects се кардинално скапи
  28. селективноста на податоците е ниска за дел од релациите
- Затоа, перформансите во вакви аналитички погледи зависат повеќе од структурата на JOIN-овите и обемот на податоци, отколку од дополнително индексирање.

Results 1 X

EXPLAIN ANALYZE SELECT \* FROM Researcher\_Access

AZ QUERY PLAN

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..0.000 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=0.153..4.369 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..0.000 rows=249 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..0.000 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..0.000 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..0.000 rows=1 loops=1)
22	Index Cond: (institution_id = ra.institution_id)
23	Planning Time: 1.091 ms
24	Execution Time: 146.949 ms

Refresh Save Cancel

Export data 200 24 24 row(s) fe

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 X

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,

Activate Go to Setting

```

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

```

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

```
JOIN Users u ON u.user_id = tsl.performed_by_user;
EXPLAIN ANALYZE SELECT * FROM Treatment_History WHERE object_id = 100;
--koj avtor koi publikaciji 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 е object\_id, со цел да се прикажат сите чекори на конзервација и третман за одреден предмет. Иницијалното време на извршување изнесуваше 0.549 ms, што е прифатливо за апликацијата. Планерот не користи дополнителни специјални индекси за овој поглед, туку се потпира на постоечките primary key индекси на табелите што учествуваат во JOIN операциите. Во тековната база не беа воведени нови индекси за овој view, бидејќи перформансите се веќе задоволителни.

```

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

```

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`.

The screenshot shows a PostgreSQL query editor with the following SQL script:

```

EXPLAIN ANALYZE SELECT * FROM Treatment_History WHERE object_id = 100;
--koj avtor kei publikaciji 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;

```

The results window shows the following query plan:

Step	Operation	Cost	Rows	Width	Actual Time	Actual Rows	Actual Loops
1	Nested Loop	(cost=1.14..21.20)	rows=1	width=112	(actual time=1158.195..1158.204)	rows=1	loops=1
2	-> Nested Loop	(cost=0.84..12.89)	rows=1	width=94	(actual time=1009.399..1009.407)	rows=1	loops=1
3	-> Index Only Scan using publication_authors_pkey on publication_authors pa	(cost=0.42..4.44)	rows=1	width=16	(actual time=0.000..0.000)	rows=1	loops=1
4	Index Cond: (publication_id = 100)						
5	Heap Fetches: 0						
6	-> Index Scan using publications_pkey on publications p	(cost=0.42..8.44)	rows=1	width=86	(actual time=498.701..498.701)	rows=1	loops=1
7	Index Cond: (publication_id = 100)						
8	-> Index Scan using authors_pkey on authors a	(cost=0.29..8.31)	rows=1	width=34	(actual time=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						

Во тековната база не се воведени нови индекси специјално за овој поглед, па планерот се потпира на постоечките primary key индекси и оптимизацијата на join операциите

По оптимизацијата планерот користи:

29. Index Only Scan using publication\_authors\_pkey
30. Index Scan using publications\_pkey
31. Index Scan using authors\_pkey
- 32.

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

```

EXPLAIN ANALYZE SELECT * FROM Treatment_History WHERE object_id = 100;
--koj avtor koi publikaciji 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;

```

Results 1

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

AZ QUERY PLAN

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

Value

Nested Loop (cost=1.14..21.20 rows=1 width=112)

Refresh Save Cancel Export data 200 11 11 row(s) fetched 5:14s on 2026-05-06 at 12

CEST en Writable Smart Insert 1443:39:49440 Set: 0 | 0

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

```

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

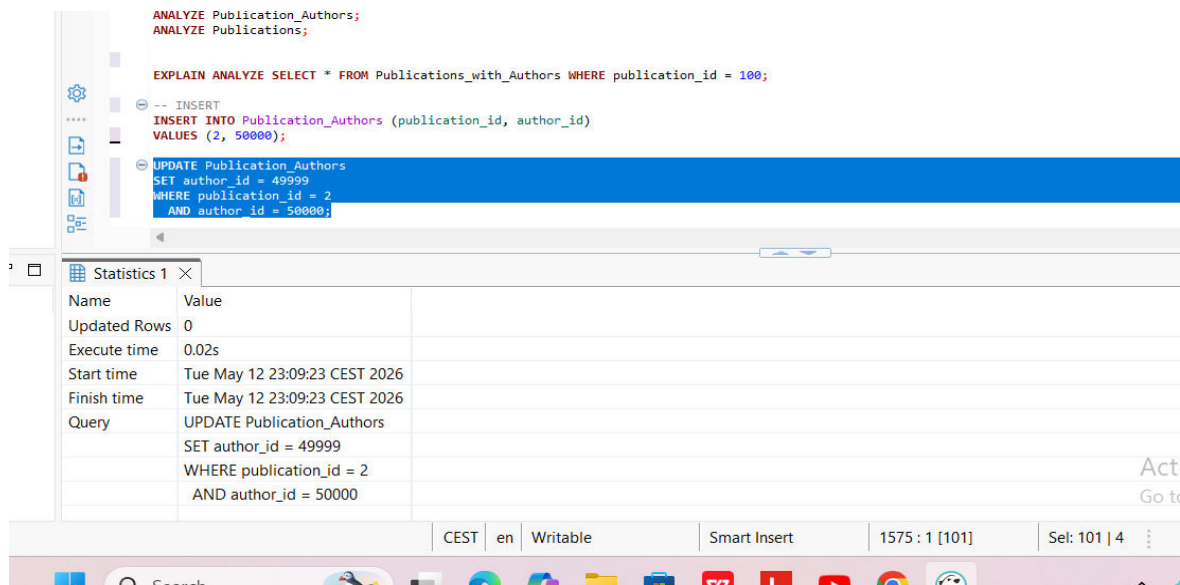
EXPLAIN ANALYZE SELECT * FROM Publications_with_Authors WHERE publication_id = 100;

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

```

Statistics 1

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)



## 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.

```

JOIN Sites s ON r.site_id = s.site_id
JOIN Regions r ON s.region_id = r.region_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;

--podeleni spored lokalitet

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

```

The screenshot shows a query execution plan for the following query: `EXPLAIN ANALYZE SELECT * FROM Heritage_Full_Overview WHERE site_id = 100 LIMIT 10;`. The plan is as follows:

- Limit (cost=1.01..1653.29 rows=10 width=347) (actual time=340.394..4055.268 rows=10 loops=1)
- > Nested Loop (cost=1.01..288158.78 rows=1744 width=347) (actual time=340.392..4055.255 rows=10 loops=1)
- > Nested Loop (cost=0.58..274274.70 rows=1744 width=331) (actual time=340.335..3096.154 rows=10 loops=1)
- > Seq Scan on fragments f (cost=0.00..274228.20 rows=1744 width=56) (actual time=340.164..3095.916 rows=10 loops=1)
  - Filter: (site\_id = 100)
  - Rows Removed by Filter: 47805
- > Materialize (cost=0.58..24.70 rows=1 width=283) (actual time=0.018..0.019 rows=1 loops=10)
- > Nested Loop (cost=0.58..24.70 rows=1 width=283) (actual time=0.159..0.165 rows=1 loops=1)
- > Nested Loop (cost=0.43..16.50 rows=1 width=173) (actual time=0.120..0.124 rows=1 loops=1)
- > 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)
  - Index Cond: (site\_id = 100)
- > 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)
  - Index Cond: (region\_id = s.region\_id)
- > 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)
  - Index Cond: (protection\_status\_id = s.protection\_status\_id)
- > 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)
  - Index Cond: (object\_id = f.object\_id)
- Planning Time: 1.215 ms
- Execution Time: 4055.439 ms

Поставени се следните индекси:

```

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

```

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

33. Index Scan using idx\_fragments\_site\_id on fragments — наместо скенирање на 10 милиони редови
34. Index Scan using objects\_pkey on objects
35. Index Scan using sites\_pkey on sites
36. Index Scan using regions\_pkey on regions
37. 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 X

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

Grid

Grid	Text
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..0.061 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=0.061..0.061 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

```
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);
```

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)

```
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
```

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

CEST en Writable Smart Insert 1238 : 10 : 434

