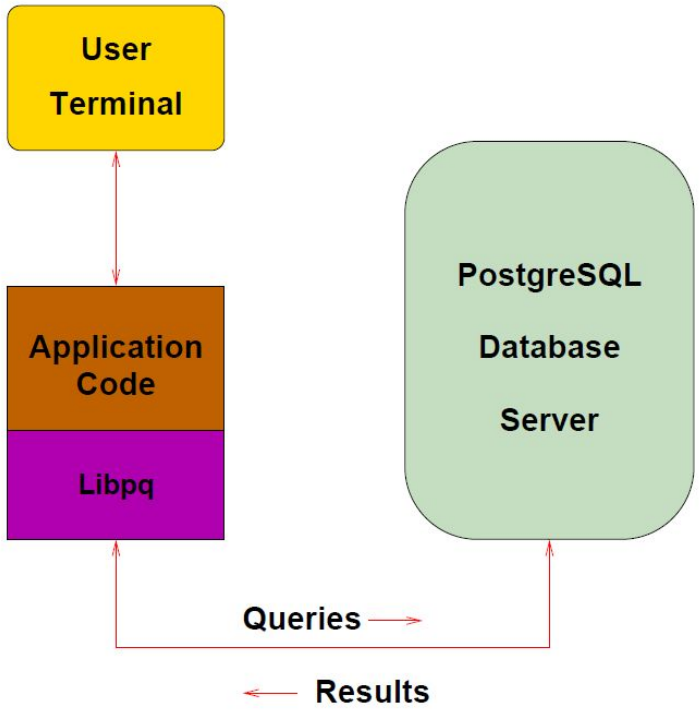
Επεξεργασία και βελτιστοποίηση αιτημάτων στην PostgreSQL

Φροντιστήριο στο μάθημα «Βάσεις Δεδομένων»

Βάσεις Δεδομένων
Πολυτεχνείο Κρήτης

Περιεχόμενα

- Ο Βελτιστοποιητής αποτελεί τον «εγκέφαλο» ενός ΣΔΒΔ δεδομένων, ερμηνεύει τις ερωτήσεις SQL και καθορίζει την ταχύτερη μέθοδο εκτέλεσής τους.
- Παρουσίαση της χρήσης των εντολών EXPLAIN και EXPLAIN ANALYSE για να δειχθεί πώς ο βελτιστοποιητής ερμηνεύει τις ερωτήσεις SQL και καθορίζει την ταχύτερη μέθοδο εκτέλεσής τους.



https://momjian.us/presentations

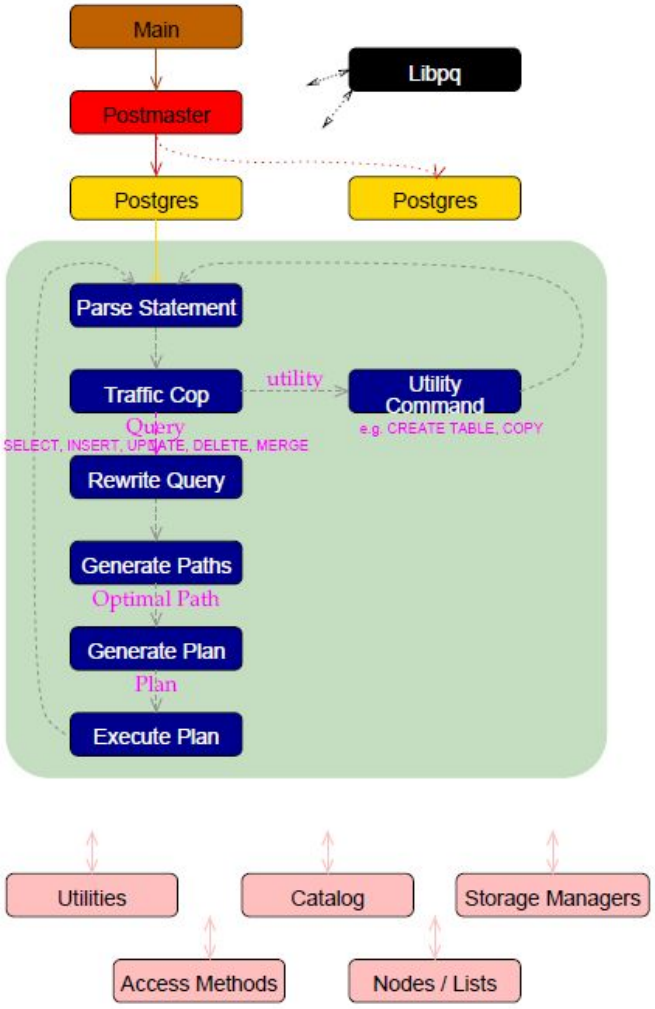
Εναλλακτικά σχέδια

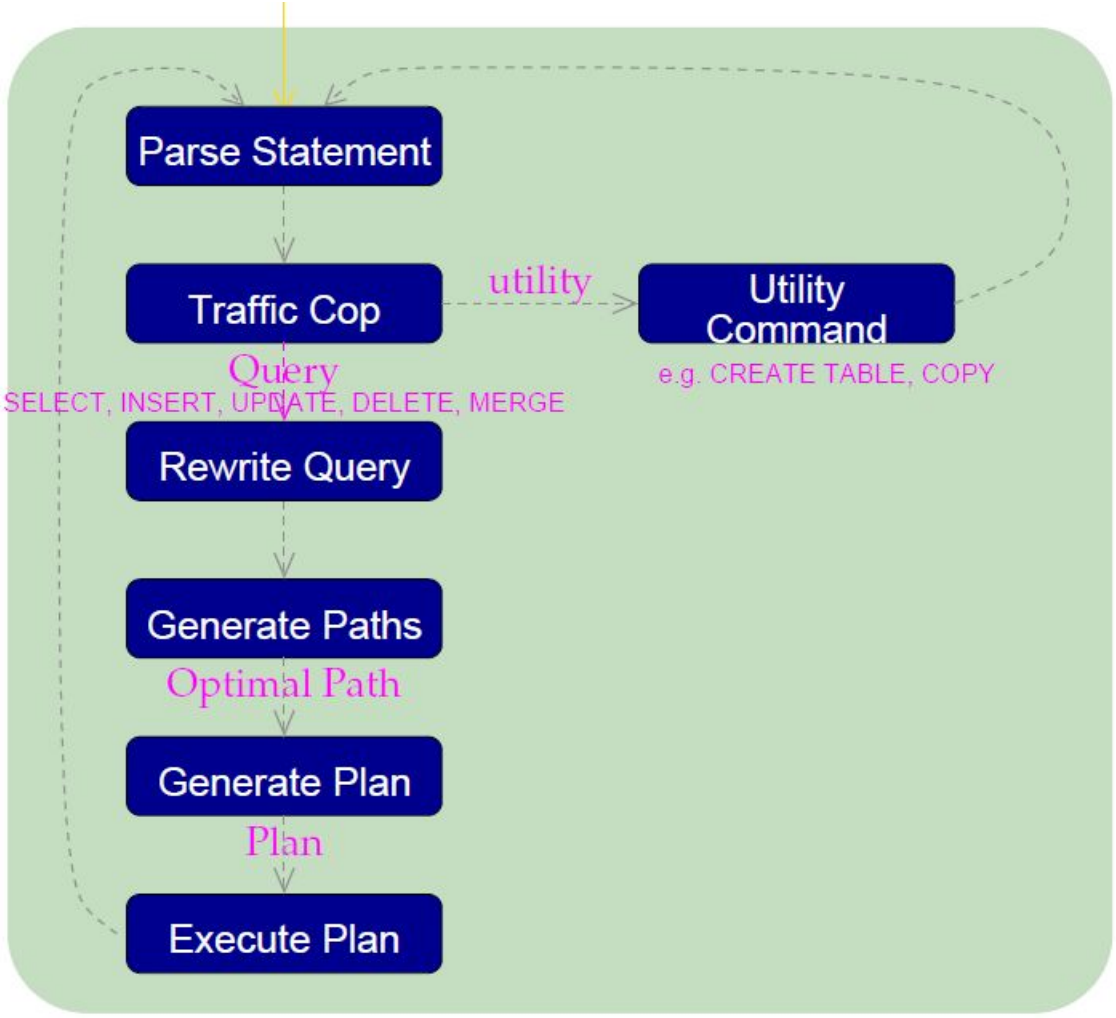
- Υπάρχουν εναλλακτικοί τρόποι επεξεργασίας κάθε αιτήματος με διαφορετικά κόστη
- Η επιλογή του τρόπου εκτέλεσης (σχέδιο - plan) με το μικρότερο κόστος είναι ένα δύσκολο πρόβλημα
- Ο αριθμός των εναλλακτικών τρόπων αυξάνει όσο αυξάνουν οι τελεστές (επί μέρους λειτουργίες) που χρησιμοποιούνται σε ένα ερώτημα
  - Μια ερώτηση είναι ουσιαστικά μια αλγεβρική παράσταση (δέντρο) που περιλαμβάνει τους σχεσιακούς τελεστές

Η λειτουργία του βελτιστοποιητή ερωτήσεων

- Αρχικά γίνεται συντακτική ανάλυση της ερώτησης
- Στη συνέχεια ο βελτιστοποιητής επιχειρεί να προσδιορίσει ένα αποδοτικό σχέδιο επεξεργασίας
  - Παραγωγή **εναλλακτικών σχεδίων**
  - **Εκτίμηση κόστους** χρησιμοποιώντας πληροφορία από τους **καταλόγους** του συστήματος
  - Επιλογή του σχεδίου που έχει το μικρότερο αναμενόμενο κόστος

Η εκτέλεση ερωτήσεων στην Postgres





https://momjian.us/main/presentations/internals.html#internal_pics

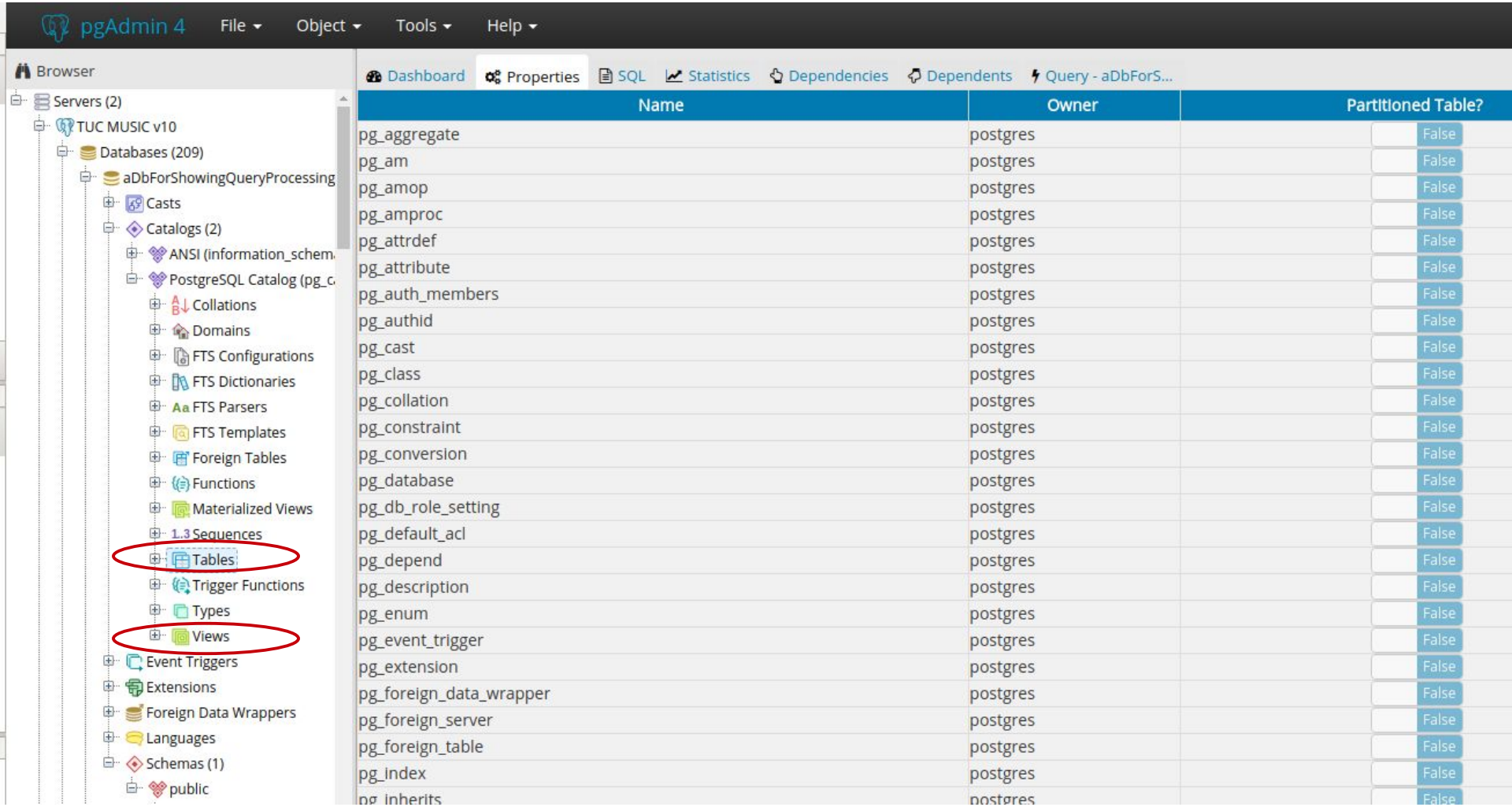
Κατάλογος συστήματος

- Καταχώρηση πληροφοριών για κάθε πίνακα και κάθε ευρετήριο καθώς και για τις όψεις, τις συναρτήσεις κλ.π.
- Συλλογή πινάκων που ονομάζονται **πίνακες του καταλόγου ή κατάλογος του συστήματος ή κατάλογος ή λεξικό δεδομένων ή μεταδεδομένα**
- Ο βελτιστοποιητής χρησιμοποιεί τον κατάλογο για να υπολογίζει το κόστος των εναλλακτικών σχεδίων καθώς και το μέγεθος των ενδιάμεσων αποτελεσμάτων.

Ο κατάλογος ως συλλογή πινάκων

- Επιτρέπεται η διαχείριση του όπως και στους πίνακες μιας βάσης που ορίζει ο χρήστης
- Ωστόσο το ΣΔΒΔ χειρίζεται με ειδικό τρόπο τους καταλόγους γνωρίζοντας το σχήμα τους.

Κατάλογος στην PostgreSQL



Η βάση για τα παραδείγματα που θα χρησιμοποιήσουμε

CITY

(*) id: integer (PK)
name: text
countrycode: character(3) (FK -> COUNTRY)
district: text
population:integer

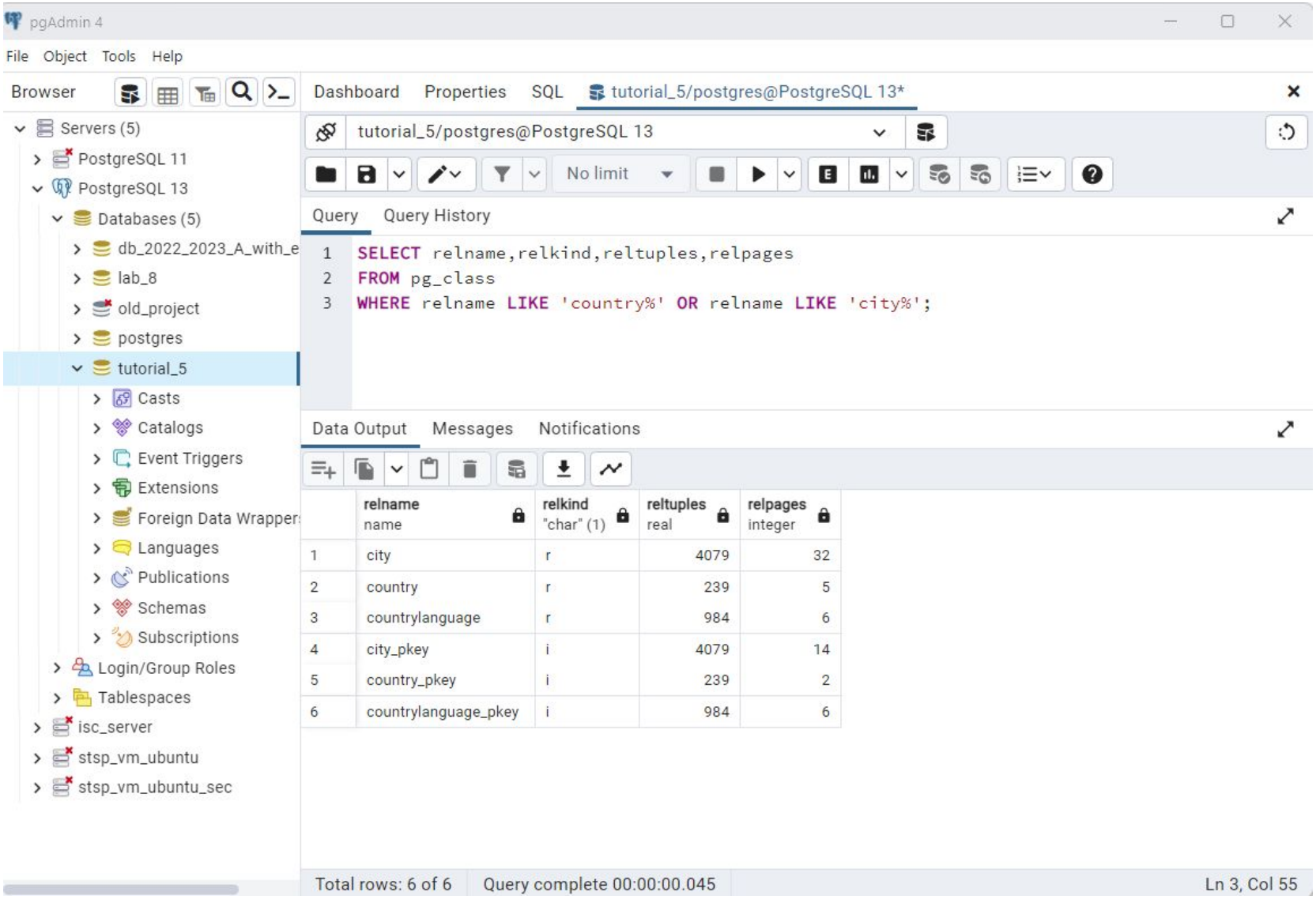
COUNTRYLANGUAGE

(*) countrycode character(3) (FK -> COUNTRY)
(*) language text
isofficial boolean
percentage real

COUNTRY

(*) code character(3)
name text
continent text
region text
surfacearea real
indepyear smallint
population integer
lifeexpectancy real
gnp numeric(10,2)
gnpold numeric(10,2)
localname text
governmentform text
headofstate text
capital integer (FK -> CITY)
code2 character

Πληροφορίες καταλόγου



Σχέδια υπολογισμού στην PostgreSQL

- Η εντολή EXPLAIN μας δείχνει το σχέδιο εκτέλεσης μιας ερώτησης
- Η εντολή EXPLAIN ANALYSE παρουσιάζει αναλυτικά τους. πραγματικούς χρόνους εκτέλεσης μιας ερώτησης σε αντιπαραβολή με τις αρχικές εκτιμήσεις κόστους του σχεδίου που εφαρμόστηκε.
- Μελετώντας τα σχέδια και τους χρόνους εκτέλεσης μπορούμε να καταλήξουμε σε αποφάσεις που αφορούν το φυσικό σχεδιασμό της βάσης και βελτιώνουν την απόδοση κρίσιμων αιτημάτων.
- Με τη βοήθεια του καταλόγου μπορούμε να εκτιμήσουμε την επιλεκτικότητα (selectivity) κάθε λογικής συνθήκης και να επιλέξουμε το/τα καλύτερο/-α από τα υποψήφια ευρετήρια.
- Μπορούμε ακόμη να δούμε αν η σειρά εκτέλεσης των συνδέσεων είναι βέλτιστη. Αν δεν είναι και θέλουμε, να επιβάλουμε μια ορισμένη σειρά εκτέλεσης των συνδέσεων.
- Αν δεν χρησιμοποιούνται ευρετήρια που υπάρχουν ίσως οφείλεται σε παρωχημένη στατιστική πληροφορία.
  - Με την εντολή VACUUM ANALYSE μπορούμε να ζητήσουμε ρητά την ανανέωση της στατιστικής πληροφορίας για όλη τη βάση ή για κάποιο πίνακα ώστε να βοηθήσουμε το βελτιστοποιητή

Η εντολή EXPLAIN

```
1  EXPLAIN SELECT country.name, city.name,
2      round(city.population*100.0/country.population,2)
3  FROM city, country
4  WHERE country.capital=city.id
5      AND country.continent='Europe';
```

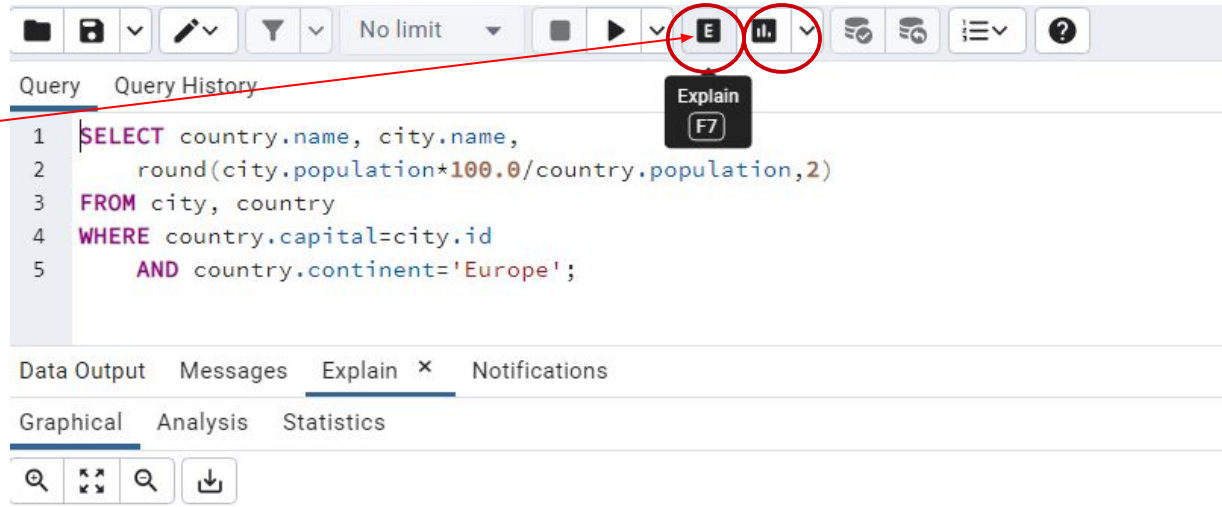
	QUERY PLAN text
1	Hash Join (cost=8.56..97.67 rows=46 width=52)
2	Hash Cond: (city.id = country.capital)
3	-> Seq Scan on city (cost=0.00..72.79 rows=4079 width=17)
4	-> Hash (cost=7.99..7.99 rows=46 width=19)
5	-> Seq Scan on country (cost=0.00..7.99 rows=46 width=19)
6	Filter: (continent = 'Europe'::text)

Για κάθε βήμα (κόμβο) του σχεδίου δίνονται **εκτιμήσεις** για:

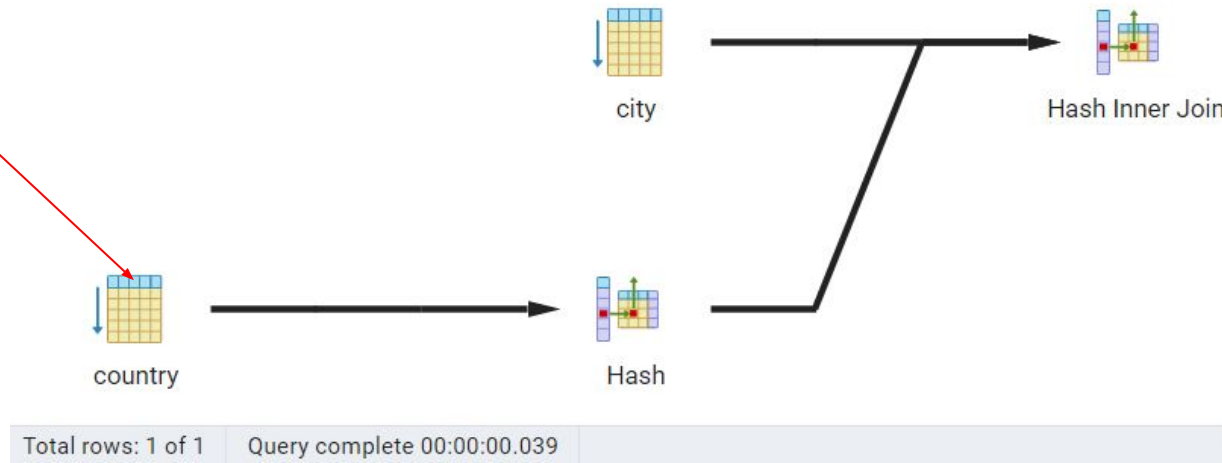
- Χρόνο υπολογισμού πρώτης πλειάδας
- Χρόνο ολοκλήρωσης υπολογισμού
- Πλήθος πλειάδων αποτελέσματος
- Μέγεθος κάθε πλειάδας του αποτελέσματος

Γραφική απεικόνιση σχεδίων

Επιλέγουμε “Explain” ή “Explain Analyse” (εναλλακτικά πατάμε F7 ή Shift-F7 αντίστοιχα)



Πατώντας με το ποντίκι πάνω σε ένα κόμβο, βλέπουμε αναλυτικές πληροφορίες



Η εντολή EXPLAIN ANALYSE

```
1  EXPLAIN ANALYSE SELECT country.name, city.name,
2      round(city.population*100.0/country.population,2)
3  FROM city, country
4  WHERE country.capital=city.id
5      AND country.continent='Europe';
```

	QUERY PLAN text
1	Hash Join (cost=8.56..97.67 rows=46 width=52) (actual time=0.083..0.581 rows=46 loops=1)
2	Hash Cond: (city.id = country.capital)
3	-> Seq Scan on city (cost=0.00..72.79 rows=4079 width=17) (actual time=0.010..0.193 rows=4079 loops=1)
4	-> Hash (cost=7.99..7.99 rows=46 width=19) (actual time=0.041..0.041 rows=46 loops=1)
5	Buckets: 1024 Batches: 1 Memory Usage: 11kB
6	-> Seq Scan on country (cost=0.00..7.99 rows=46 width=19) (actual time=0.008..0.034 rows=46 loops=1)
7	Filter: (continent = 'Europe'::text)
8	Rows Removed by Filter: 193
9	Planning Time: 0.127 ms
10	Execution Time: 0.599 ms

Παρατηρήσεις για τους χρόνους

- Οι χρόνοι είναι σε msec ενώ οι εκτιμήσεις κόστους είναι σε αφηρημένες μονάδες με αποτέλεσμα να μην έχει νόημα η αριθμητική σύγκρισή τους.
- Αυτό που έχει νόημα είναι η σύγκριση του αναμενόμενου μεγέθους του αποτελέσματος σε σχέση με το πραγματικό.
- Το loops μας δείχνει πόσες φορές εκτελείται ένας κόμβος καθώς υπάρχουν περιπτώσεις (π.χ. Nested loop join) επαναλαμβανόμενης εκτέλεσης ενός κόμβου. Τα πραγματικά κόστη σε αυτή την περίπτωση είναι μέσοι όροι.

Η εντολή EXPLAIN ANALYSE

```
1  EXPLAIN ANALYSE SELECT country.name, city.name,
2      round(city.population*100.0/country.population,2)
3  FROM city, country
4  WHERE country.capital=city.id
5      AND country.continent='Europe'
6  ORDER BY city.name;
```

	QUERY PLAN text
1	Sort (cost=98.94..99.06 rows=46 width=52) (actual time=0.571..0.573 rows=46 loops=1)
2	Sort Key: city.name
3	Sort Method: quicksort Memory: 28kB
4	-> Hash Join (cost=8.56..97.67 rows=46 width=52) (actual time=0.065..0.501 rows=46 loops=1)
5	Hash Cond: (city.id = country.capital)
6	-> Seq Scan on city (cost=0.00..72.79 rows=4079 width=17) (actual time=0.010..0.189 rows=4079 loops=1)
7	-> Hash (cost=7.99..7.99 rows=46 width=19) (actual time=0.043..0.043 rows=46 loops=1)
8	Buckets: 1024 Batches: 1 Memory Usage: 11kB
9	-> Seq Scan on country (cost=0.00..7.99 rows=46 width=19) (actual time=0.009..0.036 rows=46 loops=1)
10	Filter: (continent = 'Europe'::text)
11	Rows Removed by Filter: 193
12	Planning Time: 0.173 ms
13	Execution Time: 0.590 ms

Επιπλέον πληροφορίες

- Οι κόμβοι ταξινόμησης (Sort) δείχνουν επίσης τη μέθοδο που χρησιμοποιήθηκε και το μέγεθος της μνήμης (RAM ή δίσκου) που απαιτήθηκε.
- Οι κόμβοι κατακερματισμού (Hash) δείχνουν πόσοι κάδοι (buckets) και πόσες δέσμες (batches) απαιτήθηκαν καθώς και το μέγιστο μέγεθος κύριας μνήμης που απαιτήθηκε.
  - Αν οι δέσμες είναι περισσότερες από μία, χρησιμοποιήθηκε και χώρος στο δίσκο αλλά αυτό δεν δείχνεται.

tutorial_5/postgres@PostgreSQL 13

No limit

Query | Query History

```
1  EXPLAIN (ANALYSE,BUFFERS) SELECT country.name, city.name,
2      round(city.population*100.0/country.population,2)
3  FROM city, country
4  WHERE country.capital=city.id
5      AND country.continent='Europe'
6  ORDER BY city.name;
```

Data Output | Messages | Explain | Notifications

	QUERY PLAN text
1	Sort (cost=98.94..99.06 rows=46 width=52) (actual time=0.555..0.557 rows=46 loops=1)
2	Sort Key: city.name
3	Sort Method: quicksort Memory: 28kB
4	Buffers: shared hit=37
5	-> Hash Join (cost=8.56..97.67 rows=46 width=52) (actual time=0.062..0.486 rows=46 loops=1)
6	Hash Cond: (city.id = country.capital)
7	Buffers: shared hit=37
8	-> Seq Scan on city (cost=0.00..72.79 rows=4079 width=17) (actual time=0.011..0.182 rows=4079 loops...
9	Buffers: shared hit=32
10	-> Hash (cost=7.99..7.99 rows=46 width=19) (actual time=0.043..0.043 rows=46 loops=1)
11	Buckets: 1024 Batches: 1 Memory Usage: 11kB
12	Buffers: shared hit=5
13	-> Seq Scan on country (cost=0.00..7.99 rows=46 width=19) (actual time=0.009..0.033 rows=46 loops...
14	Filter: (continent = 'Europe'::text)
15	Rows Removed by Filter: 193
16	Buffers: shared hit=5
17	Planning:
18	Buffers: shared hit=6
19	Planning Time: 0.170 ms
20	Execution Time: 0.579 ms

Total rows: 20 of 20 | Query complete 00:00:00.045

Η επιλογή BUFFERS

Βοηθά στον εντοπισμό των σημείων που έχουν τα μεγαλύτερα κόστη Εισόδου/Εξόδου

Τι αποφάσεις χρειάζεται να πάρει ο βελτιστοποιητής;

- Μέθοδο σάρωσης (scan method)
- Μέθοδο σύνδεσης (join method)
- Σειρά σύνδεσης (join order)

Τι αποφάσεις χρειάζεται να πάρει ο βελτιστοποιητής;

- Μέθοδοι σάρωσης στην Postgres:
  - Sequential Scan
  - Bitmap Index Scan
  - Index Scan
- Μέθοδοι σύνδεσης στην Postgres:
  - Nested Loop
    - With Inner Sequential Scan
    - With Inner Index Scan
  - Hash Join
  - Merge Join

Ένα παράδειγμα για την επιλογή μεθόδου σάρωσης στην Postgres

Βασισμένο στην παρουσίαση «Explaining the Postgres Query Optimizer» του Bruce Momjian, https://momjian.us/presentations

- Δημιουργία ενός temporary table με ένα index

Query | Query History

```
1  CREATE TEMPORARY TABLE sample (form, junk) AS
2          SELECT governmentform, repeat('x', 250)
3          FROM country
4          ORDER BY random();  -- add rows in random order
5
6  CREATE INDEX i_sample on sample (form);
```

- Δημιουργία μιας EXPLAIN function

```
8  CREATE OR REPLACE FUNCTION lookup_form(text) RETURNS SETOF
9  text AS $$
10 BEGIN
11 RETURN QUERY EXECUTE '
12          EXPLAIN SELECT form
13          FROM sample
14          WHERE form = ''' || $1 || '''';
15 END
```

Ένα παράδειγμα για την επιλογή μεθόδου σάρωσης στην Postgres

- Εύρεση της κατανομής των δεδομένων στο *sample* table

```
18  WITH forms (form, count) AS (
19          SELECT  form, COUNT(*)
20          FROM sample
21          GROUP BY 1
22  )
23  SELECT form, count, (count * 100.0 / (SUM(count) OVER
24  ()))::numeric(4,1) AS "%"
25  FROM forms
26  ORDER BY 2 DESC;
```

Ένα παράδειγμα για την επιλογή μεθόδου σάρωσης στην Postgres

- Εύρεση της κατανομής των δεδομένων στο *sample* table

	form text	count bigint	% numeric (4,1)
1	Republic	122	51.0
2	Constitutional Monarchy	29	12.1
3	Federal Republic	15	6.3
4	Dependent Territory of the UK	12	5.0
5	Monarchy	5	2.1
6	Overseas Department of Fra...	4	1.7
7	Territory of Australia	4	1.7
8	Constitutional Monarchy, Fed...	4	1.7
9	Nonmetropolitan Territory of ...	4	1.7
10	Socialistic Republic	3	1.3
11	Nonmetropolitan Territory of ...	3	1.3
12	US Territory	3	1.3
13	Part of Denmark	2	0.8
14	Monarchy (Sultanate)	2	0.8
15	Islamic Republic	2	0.8

...

20	Nonmetropolitan Territory of ...	2	0.8
21	Administrated by the UN	1	0.4
22	Parlementary Monarchy	1	0.4
23	Emirate Federation	1	0.4
24	Parliamentary Coprincipality	1	0.4
25	Monarchy (Emirate)	1	0.4
26	Socialistic State	1	0.4
27	Co-administrated	1	0.4
28	Independent Church State	1	0.4
29	Occupied by Marocco	1	0.4
30	Federation	1	0.4
31	Constitutional Monarchy (Em...	1	0.4
32	Autonomous Area	1	0.4
33	People'sRepublic	1	0.4
34	Islamic Emirate	1	0.4
35	Dependent Territory of the US	1	0.4

Total rows: 35 of 35 | Query complete 00:00:00.064 | Rows

Ένα παράδειγμα για την επιλογή μεθόδου σάρωσης στην Postgres

- Πόσο σημαντική είναι η κατανομή των δεδομένων;

```
30  EXPLAIN SELECT form
31  FROM sample
32  WHERE form = 'Republic';
```

	QUERY PLAN text
1	Index Only Scan using i_sample on sample (cost=0.14..8.16 rows=1 width=32)
2	Index Cond: (form = 'Republic'::text)

```
34  EXPLAIN SELECT form
35  FROM sample
36  WHERE form = 'Monarchy';
```

	QUERY PLAN text
1	Index Only Scan using i_sample on sample (cost=0.14..8.16 rows=1 width=32)
2	Index Cond: (form = 'Monarchy'::text)

```
38  EXPLAIN SELECT form
39  FROM sample
40  WHERE form = 'Federation';
```

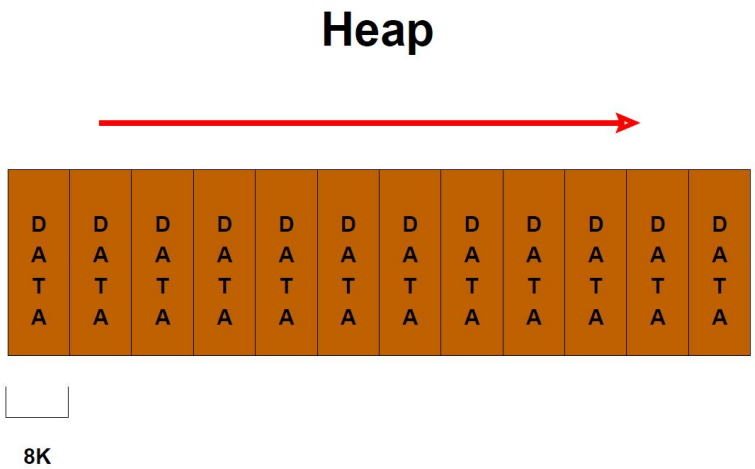
	QUERY PLAN text
1	Index Only Scan using i_sample on sample (cost=0.14..8.16 rows=1 width=32)
2	Index Cond: (form = 'Federation'::text)

Ένα παράδειγμα για την επιλογή μεθόδου σάρωσης στην Postgres

- Η εκτέλεση της ANALYZE προκαλεί sequential scan για τις πιο συχνές τιμές

```
40  ANALYZE sample;
41
42  EXPLAIN SELECT form
43  FROM sample
44  WHERE form = 'Republic';
```

	QUERY PLAN text
1	Seq Scan on sample (cost=0.00..12.99 rows=122 width=16)
2	Filter: (form = 'Republic'::text)



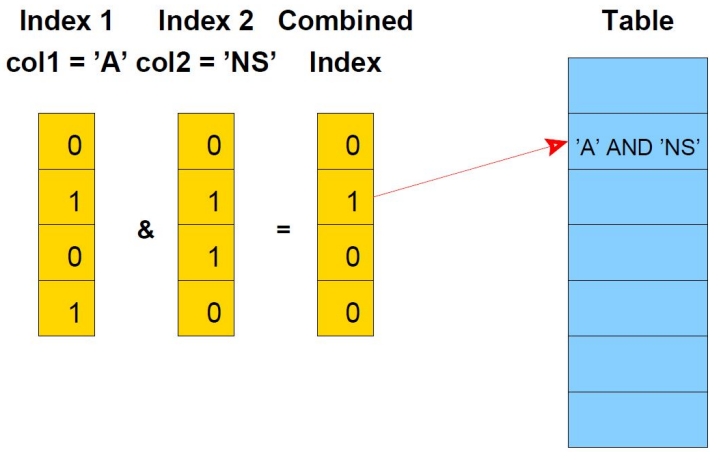
- Η ANALYZE δεν εκτελείται αυτόματα στους προσωρινούς πίνακες γιατί αυτοί είναι ορατοί μόνο στη σύνοδο που τους δημιούργησε

Ένα παράδειγμα για την επιλογή μεθόδου σάρωσης στην Postgres

- Η εκτέλεση της ANALYZE προκαλεί bitmap index scan για τις λιγότερο συχνές τιμές

```
46  EXPLAIN SELECT form
47  FROM sample
48  WHERE form = 'Monarchy';
```

	QUERY PLAN text
1	Bitmap Heap Scan on sample (cost=4.18..12.66 rows=5 width=16)
2	Recheck Cond: (form = 'Monarchy'::text)
3	-> Bitmap Index Scan on i_sample (cost=0.00..4.18 rows=5 width=0)
4	Index Cond: (form = 'Monarchy'::text)



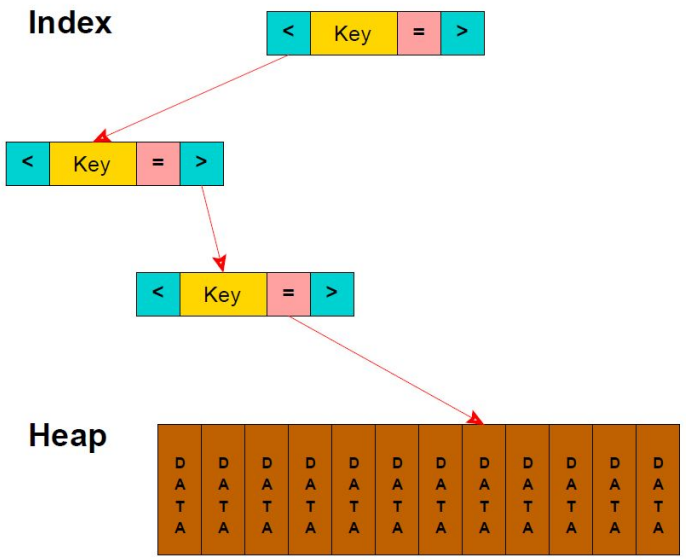
Ένα παράδειγμα για την επιλογή μεθόδου σάρωσης στην Postgres

- Η εκτέλεση της ANALYZE προκαλεί index scan για τις σπάνιες τιμές

```
50  EXPLAIN SELECT form
51  FROM sample
52  WHERE form = 'Federation';
```

	QUERY PLAN text
1	Index Only Scan using i_sample on sample (cost=0.14..8.16 rows=1 width=16)
2	Index Cond: (form = 'Federation'::text)

Επιλογή Index/Bitmap Index scan όταν η επιλεκτικότητα είναι περίπου 5-10%



Ένα παράδειγμα για την επιλογή μεθόδου σάρωσης στην Postgres

- Όλες οι τιμές…

```
54  WITH form (form, count) AS (
55          SELECT form, COUNT(*)
56          FROM sample
57          GROUP BY 1
58  )
59  SELECT form AS l, count, lookup_form(form)
60  FROM form
61  ORDER BY 2 DESC;
```

	l text	count bigint	lookup_form text
1	Republic	122	Seq Scan on sample (cost=0.00..12.99 rows=122 width=16)
2	Republic	122	Filter: (form = 'Republic'::text)
3	Constitutional Monarchy	29	Seq Scan on sample (cost=0.00..12.99 rows=29 width=16)
4	Constitutional Monarchy	29	Filter: (form = 'Constitutional Monarchy'::text)
5	Federal Republic	15	Seq Scan on sample (cost=0.00..12.99 rows=15 width=16)
6	Federal Republic	15	Filter: (form = 'Federal Republic'::text)
7	Dependent Territory of the…	12	Seq Scan on sample (cost=0.00..12.99 rows=12 width=16)
8	Dependent Territory of the…	12	Filter: (form = 'Dependent Territory of the UK'::text)
9	Monarchy	5	Bitmap Heap Scan on sample (cost=4.18..12.66 rows=5 width=16)
10	Monarchy	5	Recheck Cond: (form = 'Monarchy'::text)
11	Monarchy	5	-> Bitmap Index Scan on i_sample (cost=0.00..4.18 rows=5 width=0)
12	Monarchy	5	Index Cond: (form = 'Monarchy'::text)
13	Territory of Australia	4	Bitmap Heap Scan on sample (cost=4.18..12.64 rows=4 width=16)
14	Territory of Australia	4	Recheck Cond: (form = 'Territory of Australia'::text)
15	Territory of Australia	4	-> Bitmap Index Scan on i_sample (cost=0.00..4.17 rows=4 width=0)

Ένα παράδειγμα για την επιλογή μεθόδου σάρωσης στην Postgres

- Παίρνοντας μόνο την πρώτη γραμμή…

```
64  WITH form (form, count) AS (
65          SELECT form, COUNT(*)
66          FROM sample
67          GROUP BY 1
68  )
69  SELECT form AS l, count,
70          (SELECT *
71           FROM lookup_form(form) AS l2
72           LIMIT 1) AS lookup_form
73  FROM form
74  ORDER BY 2 DESC;
```

Ένα παράδειγμα για την επιλογή μεθόδου σάρωσης στην Postgres

	l text	count bigint	lookup_form text
1	Republic	122	Seq Scan on sample (cost=0.00..12.99 rows=122 width=16)
2	Constitutional Monarchy	29	Seq Scan on sample (cost=0.00..12.99 rows=29 width=16)
3	Federal Republic	15	Seq Scan on sample (cost=0.00..12.99 rows=15 width=16)
4	Dependent Territory of the UK	12	Seq Scan on sample (cost=0.00..12.99 rows=12 width=16)
5	Monarchy	5	Bitmap Heap Scan on sample (cost=4.18..12.66 rows=5 width=16)
6	Territory of Australia	4	Bitmap Heap Scan on sample (cost=4.18..12.64 rows=4 width=16)
7	Nonmetropolitan Territory of ...	4	Bitmap Heap Scan on sample (cost=4.18..12.64 rows=4 width=16)
8	Overseas Department of Fran...	4	Bitmap Heap Scan on sample (cost=4.18..12.64 rows=4 width=16)
9	Constitutional Monarchy, Fed...	4	Bitmap Heap Scan on sample (cost=4.18..12.64 rows=4 width=16)
10	US Territory	3	Bitmap Heap Scan on sample (cost=4.17..11.28 rows=3 width=16)
11	Nonmetropolitan Territory of ...	3	Bitmap Heap Scan on sample (cost=4.17..11.28 rows=3 width=16)
12	Socialistic Republic	3	Bitmap Heap Scan on sample (cost=4.17..11.28 rows=3 width=16)
13	Commonwealth of the US	2	Bitmap Heap Scan on sample (cost=4.16..9.50 rows=2 width=16)
14	Part of Denmark	2	Bitmap Heap Scan on sample (cost=4.16..9.50 rows=2 width=16)
15	Islamic Republic	2	Bitmap Heap Scan on sample (cost=4.16..9.50 rows=2 width=16)
16	Monarchy (Sultanate)	2	Bitmap Heap Scan on sample (cost=4.16..9.50 rows=2 width=16)
17	Special Administrative Regio...	2	Bitmap Heap Scan on sample (cost=4.16..9.50 rows=2 width=16)
18	Dependent Territory of Norway	2	Bitmap Heap Scan on sample (cost=4.16..9.50 rows=2 width=16)
19	Nonmetropolitan Territory of ...	2	Bitmap Heap Scan on sample (cost=4.16..9.50 rows=2 width=16)
20	Territorial Collectivity of Fran...	2	Bitmap Heap Scan on sample (cost=4.16..9.50 rows=2 width=16)

Πολύ συχνές τιμές

Λιγότερο συχνές τιμές

Ένα παράδειγμα για την επιλογή μεθόδου σάρωσης στην Postgres

21	Peoples Republic	1	Index Only Scan using i_sample on sample (cost=0.14..8.16 rows=1 width=...
22	Co-administrated	1	Index Only Scan using i_sample on sample (cost=0.14..8.16 rows=1 width=...
23	Parlementary Monarchy	1	Index Only Scan using i_sample on sample (cost=0.14..8.16 rows=1 width=...
24	Administrated by the UN	1	Index Only Scan using i_sample on sample (cost=0.14..8.16 rows=1 width=...
25	Federation	1	Index Only Scan using i_sample on sample (cost=0.14..8.16 rows=1 width=...
26	Emirate Federation	1	Index Only Scan using i_sample on sample (cost=0.14..8.16 rows=1 width=...
27	Constitutional Monarchy (Em...	1	Index Only Scan using i_sample on sample (cost=0.14..8.16 rows=1 width=...
28	Parliamentary Coprincipality	1	Index Only Scan using i_sample on sample (cost=0.14..8.16 rows=1 width=...
29	Autonomous Area	1	Index Only Scan using i_sample on sample (cost=0.14..8.16 rows=1 width=...
30	Monarchy (Emirate)	1	Index Only Scan using i_sample on sample (cost=0.14..8.16 rows=1 width=...
31	Socialistic State	1	Index Only Scan using i_sample on sample (cost=0.14..8.16 rows=1 width=...
32	Islamic Emirate	1	Index Only Scan using i_sample on sample (cost=0.14..8.16 rows=1 width=...
33	Independent Church State	1	Index Only Scan using i_sample on sample (cost=0.14..8.16 rows=1 width=...
34	Dependent Territory of the US	1	Index Only Scan using i_sample on sample (cost=0.14..8.16 rows=1 width=...
35	Occupied by Marocco	1	Index Only Scan using i_sample on sample (cost=0.14..8.16 rows=1 width=...

Total rows: 35 of 35 Query complete 00:00:00.071

Σπάνιες τιμές

Τα αποτελέσματα μπορεί να διαφέρουν ανάλογα με το clustering των τιμών στα heap pages

Ένα παράδειγμα για την επιλογή μεθόδου σάρωσης στην Postgres

- Επιβάλλοντας το index scan…

```
76  SET enable_seqscan = false;
77
78  SET enable_bitmapscan = false;
79
80  WITH form (form, count) AS (
81          SELECT form, COUNT(*)
82          FROM sample
83          GROUP BY 1
84  )
85  SELECT form AS l, count,
86          (SELECT *
87           FROM lookup_form(form) AS l2
88           LIMIT 1) AS lookup_form
89  FROM form
90  ORDER BY 2 DESC;
```

Ένα παράδειγμα για την επιλογή μεθόδου σάρωσης στην Postgres

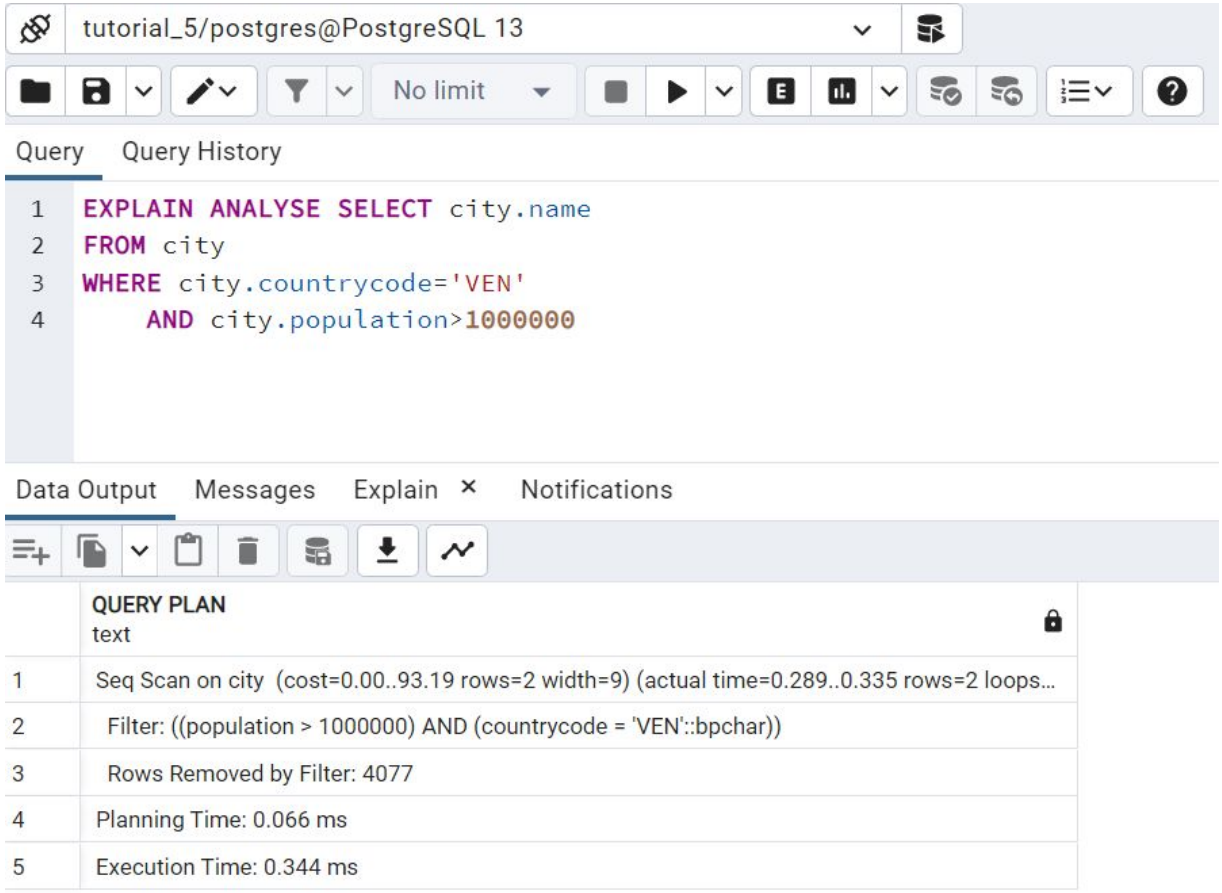
- Επιβάλλοντας το index scan…

Ήταν ~13

	l text	count bigint	lookup_form text
1	Republic	122	Index Only Scan using i_sample on sample (cost=0.14..47.58 rows=122 width=16)
2	Constitutional Monarchy	29	Index Only Scan using i_sample on sample (cost=0.14..41.60 rows=29 width=16)
3	Federal Republic	15	Index Only Scan using i_sample on sample (cost=0.14..37.62 rows=15 width=16)
4	Dependent Territory of the …	12	Index Only Scan using i_sample on sample (cost=0.14..33.92 rows=12 width=16)
5	Monarchy	5	Index Only Scan using i_sample on sample (cost=0.14..19.19 rows=5 width=16)
6	Nonmetropolitan Territory o…	4	Index Only Scan using i_sample on sample (cost=0.14..19.17 rows=4 width=16)
7	Overseas Department of Fr…	4	Index Only Scan using i_sample on sample (cost=0.14..19.17 rows=4 width=16)
8	Constitutional Monarchy, F…	4	Index Only Scan using i_sample on sample (cost=0.14..19.17 rows=4 width=16)
9	Territory of Australia	4	Index Only Scan using i_sample on sample (cost=0.14..19.17 rows=4 width=16)
10	Socialistic Republic	3	Index Only Scan using i_sample on sample (cost=0.14..15.50 rows=3 width=16)
11	Nonmetropolitan Territory o…	3	Index Only Scan using i_sample on sample (cost=0.14..15.50 rows=3 width=16)
12	US Territory	3	Index Only Scan using i_sample on sample (cost=0.14..15.50 rows=3 width=16)
13	Territorial Collectivity of Fra…	2	Index Only Scan using i_sample on sample (cost=0.14..11.83 rows=2 width=16)
14	Monarchy (Sultanate)	2	Index Only Scan using i_sample on sample (cost=0.14..11.83 rows=2 width=16)
15	Dependent Territory of Nor…	2	Index Only Scan using i_sample on sample (cost=0.14..11.83 rows=2 width=16)

Βελτίωση απόδοσης απλών ερωτήσεων

- Ποιες είναι οι πόλεις της Βενεζουέλας με πληθυσμό άνω του 1.000.000;



Χωρίς ευρετήριο αλλά με χρήση στατιστικών

Δενδρικό ευρετήριο στο population

- Ευρετήριο κατακερματισμού δεν θα χρησίμευε. (Γιατί;)

tutorial_5/postgres@PostgreSQL 13

No limit

Query | Query History

```
1  CREATE INDEX city_population_idx ON city
2      USING btree(population);
3
4  EXPLAIN ANALYSE SELECT city.name
5  FROM city
6  WHERE city.countrycode='VEN'
7      AND city.population>1000000;
```

Data Output | Messages | Explain | Notifications

	QUERY PLAN text
1	Bitmap Heap Scan on city (cost=6.04..41.55 rows=2 width=9) (actual time=0.082..0.087 rows=2 loops=1)
2	Recheck Cond: (population > 1000000)
3	Filter: (countrycode = 'VEN'::bpchar)
4	Rows Removed by Filter: 235
5	Heap Blocks: exact=25
6	-> Bitmap Index Scan on city_population_idx (cost=0.00..6.04 rows=234 width=0) (actual time=0.036..0.037 rows=237 loop...
7	Index Cond: (population > 1000000)
8	Planning Time: 0.221 ms
9	Execution Time: 0.108 ms

Με δενδρικό ευρετήριο και με χρήση στατιστικών

Ευρετήριο κατακερμ. στο countrycode

- Καλύτερο από δενδρικό για την ερώτηση αυτή (Γιατί;)

tutorial_5/postgres@PostgreSQL 13

No limit

Query | Query History

```
1  DROP INDEX city_population_idx;
2  CREATE INDEX city_coutnry_idx ON city
3      USING hash(countrycode);
4  EXPLAIN ANALYSE SELECT city.name
5  FROM city
6  WHERE city.countrycode='VEN'
7      AND city.population>1000000;
```

Data Output | Messages | Explain | Notifications

	QUERY PLAN text
1	Bitmap Heap Scan on city (cost=4.31..38.63 rows=2 width=9) (actual time=0.014..0.018 rows=2 loops=1)
2	Recheck Cond: (countrycode = 'VEN'::bpchar)
3	Filter: (population > 1000000)
4	Rows Removed by Filter: 39
5	Heap Blocks: exact=1
6	-> Bitmap Index Scan on city_coutnry_idx (cost=0.00..4.31 rows=41 width=0) (actual time=0.009..0.009 rows=41 loop...
7	Index Cond: (countrycode = 'VEN'::bpchar)
8	Planning Time: 0.163 ms
9	Execution Time: 0.032 ms

Με hash ευρετήριο και με χρήση στατιστικών

Επιπλέον εναλλακτικές...

- **Συνδυασμός ευρετηρίων**
  Χρήση και του δενδρικού ευρετηρίου στο population και του ευρετηρίου κατακερματισμού στο countrycode
- **Ευρετήριων πολλών χαρακτηριστικών**

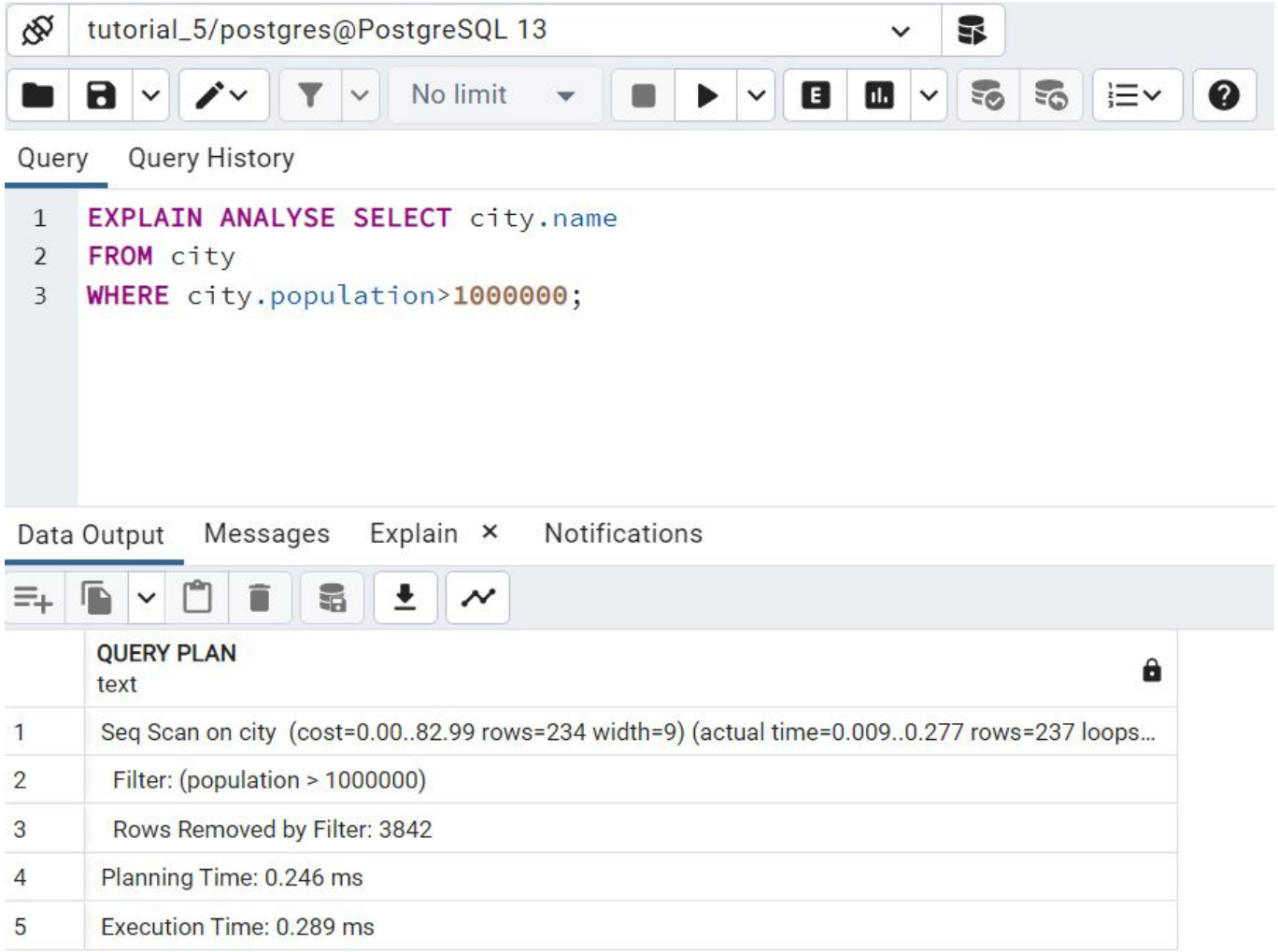
Χρήση δενδρικού (γιατί;) ευρετηρίου στα δύο χαρακτηριστικά countrycode και population (ποιο πρώτο και ποιο δεύτερο;)

- **Clustering**

Προσοχή: Μόνο ένα ευρετήριο μπορεί να χρησιμοποιηθεί για την ομαδοποίηση των πλειάδων ενός πίνακα.

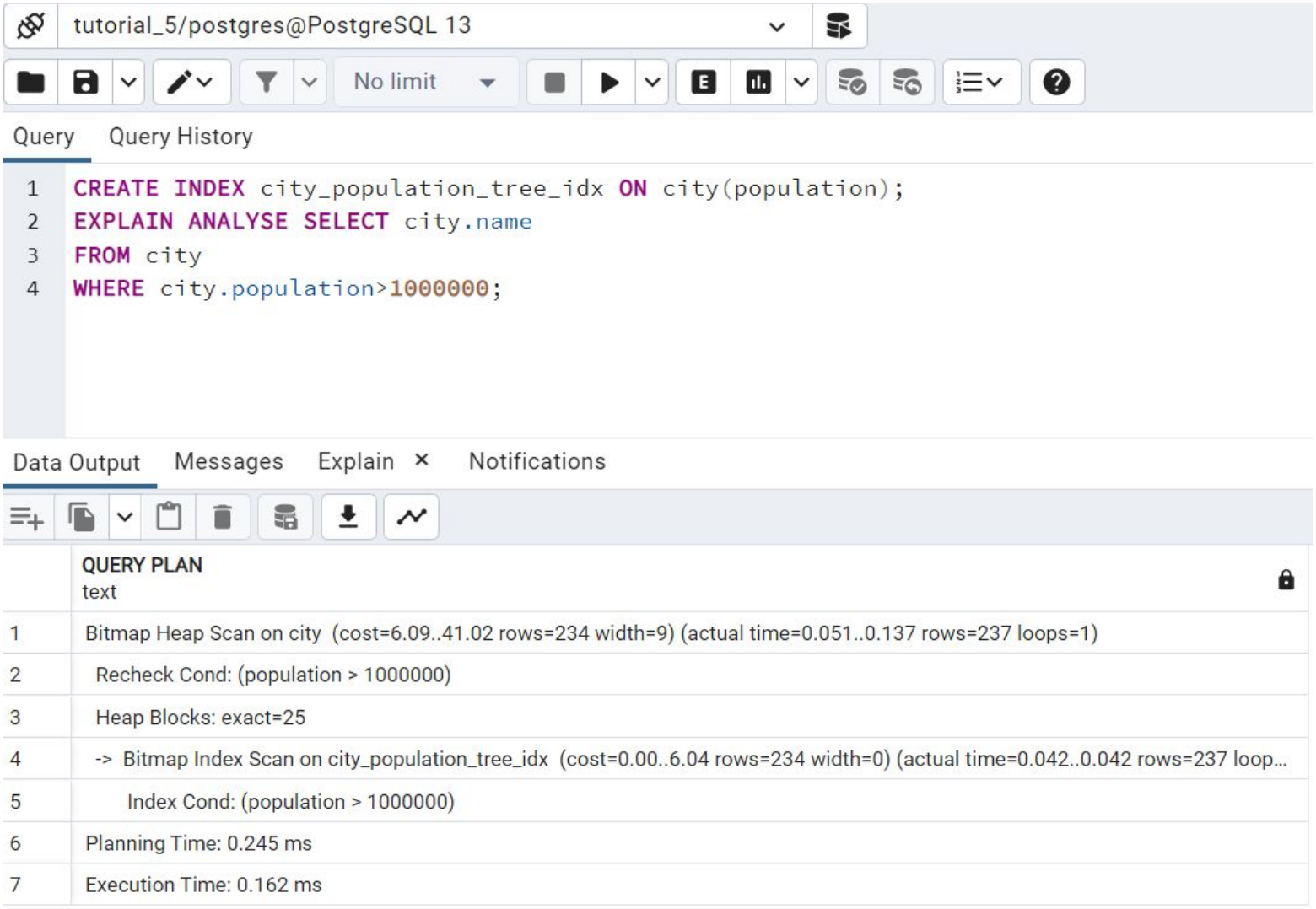
Ομαδοποίηση (Clustering) (1/3)

- Θέλουμε να βρούμε πόλεις με πληθυσμό άνω του ενός εκατομμυρίου.



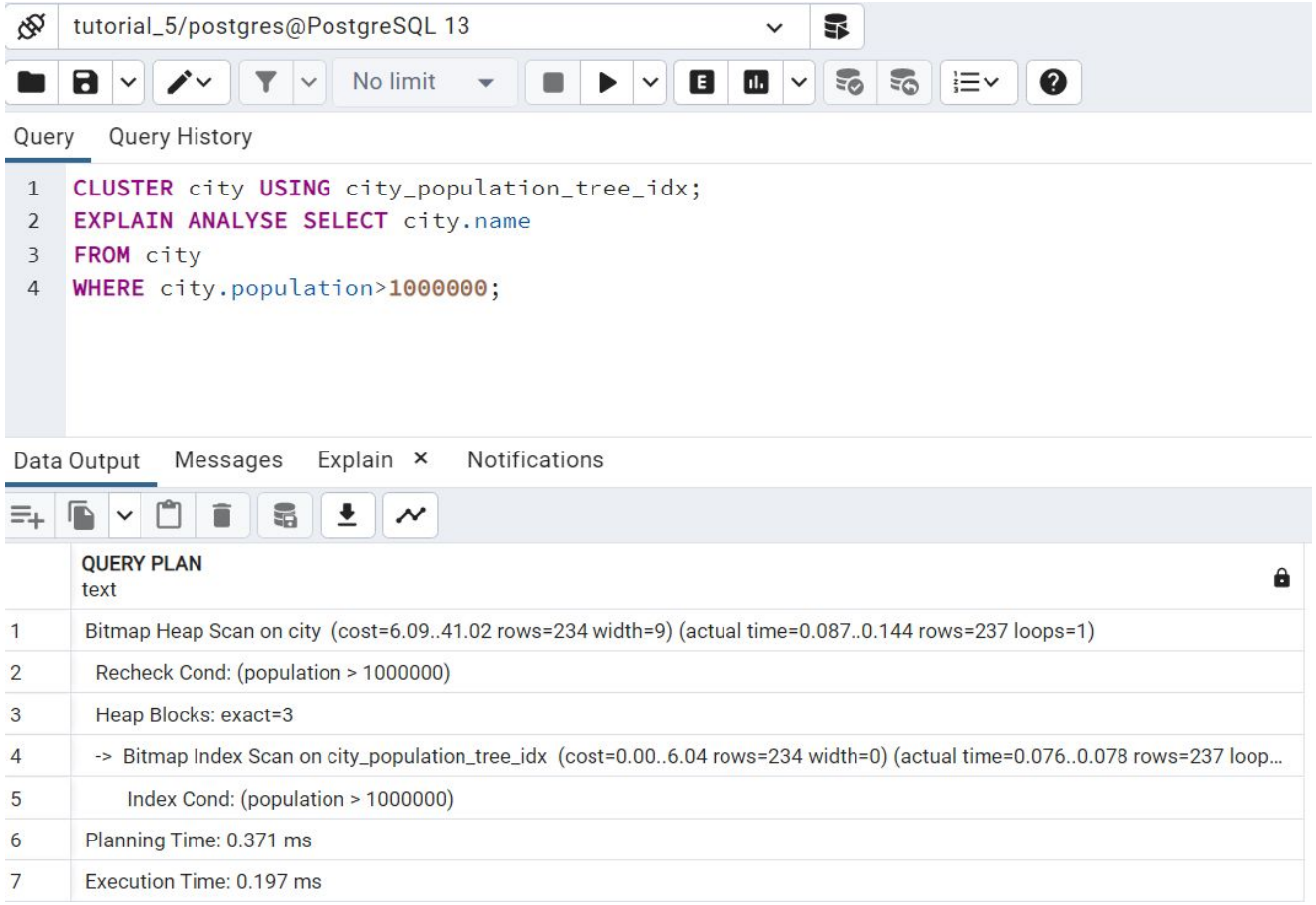
Ομαδοποίηση (Clustering) (2/3)

- Δημιουργούμε ένα δενδρικό ευρετήριο στο population



Ομαδοποίηση (Clustering) (3/3)

- Ομαδοποιούμε (clustering) τις εγγραφές του πίνακα city με βάση το ευρετήριο που μόλις δημιουργήσαμε.



- Τι παρατηρείτε; Αλλάζει το σχέδιο; Τι αλλάζει; Γιατί;

Ένα παράδειγμα για την επιλογή μεθόδου σύνδεσης στην Postgres

Βασισμένο στην παρουσίαση «Explaining the Postgres Query Optimizer» του Bruce Momjian, https://momjian.us/presentations

- Δημιουργία 2 temporary tables (χωρίς index και χωρίς στατιστικά)

```
1  CREATE TEMPORARY TABLE sample1 (id, junk) AS
2      SELECT city.countrycode, repeat('x',250)
3      FROM city
4      ORDER BY RANDOM();
5
6  CREATE TEMPORARY TABLE sample2 (id, junk) AS
7      SELECT country.code, repeat('x',250)
8      FROM country
9      ORDER BY RANDOM();
```

Ένα παράδειγμα για την επιλογή μεθόδου σύνδεσης στην Postgres

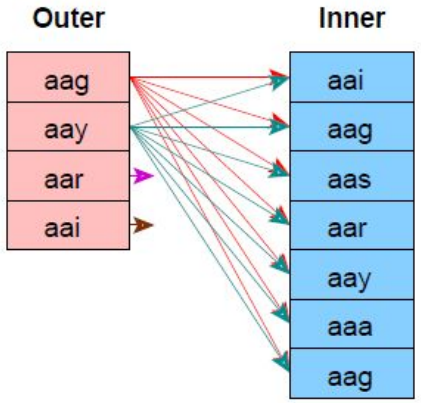
- Σύνδεση των δύο tables με αυστηρή συνθήκη επιλογής

```
11  EXPLAIN SELECT sample2.junk
12  FROM sample1 JOIN sample2 ON (sample1.id = sample2.id)
13  WHERE sample1.id = 'BRA'
```

	QUERY PLAN text
1	Nested Loop (cost=0.00..383.75 rows=405 width=32)
2	-> Seq Scan on sample1 (cost=0.00..355.30 rows=81 width=16)
3	Filter: (id = 'BRA'::bpchar)
4	-> Materialize (cost=0.00..23.40 rows=5 width=48)
5	-> Seq Scan on sample2 (cost=0.00..23.38 rows=5 width=48)
6	Filter: (id = 'BRA'::bpchar)

Επιλογή nested loop join για φόρτους μικρού μεγέθους

Nested Loop Join with Inner Sequential Scan



No Setup Required

Used For Small Tables

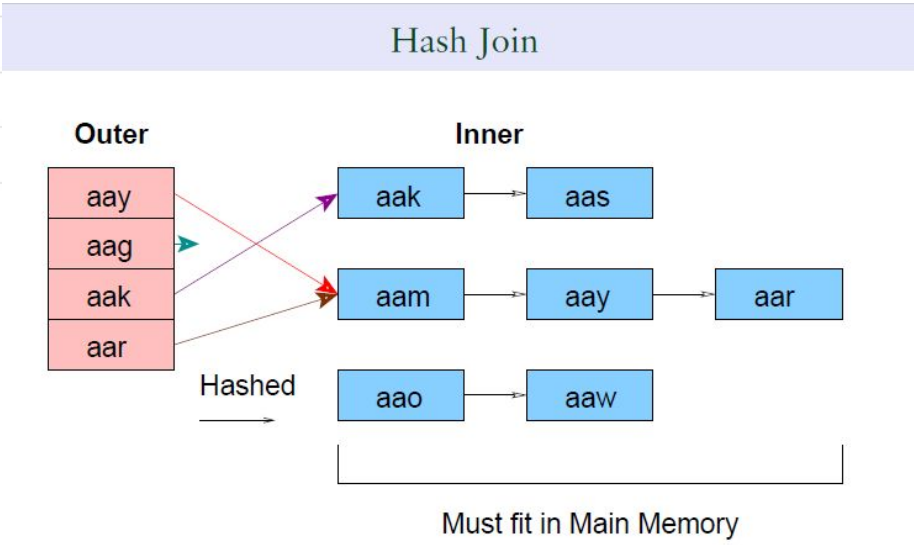
Ένα παράδειγμα για την επιλογή μεθόδου σύνδεσης στην Postgres

- Σύνδεση των δύο tables με πιο χαλαρή συνθήκη επιλογής

```
15  EXPLAIN SELECT sample2.junk
16  FROM sample1 JOIN sample2 ON (sample1.id = sample2.id)
17  WHERE sample1.id > 'BOL' and sample1.id < 'BRA'
```

	QUERY PLAN text
1	Hash Join (cost=396.97..435.38 rows=433 width=32)
2	Hash Cond: (sample2.id = sample1.id)
3	-> Seq Scan on sample2 (cost=0.00..20.70 rows=1070 width=48)
4	-> Hash (cost=395.96..395.96 rows=81 width=16)
5	-> Seq Scan on sample1 (cost=0.00..395.96 rows=81 width=16)
6	Filter: ((id > 'BOL'::bpchar) AND (id < 'BRA'::bpchar))

Επιλογή hash join για φόρτους μεσαίου μεγέθους

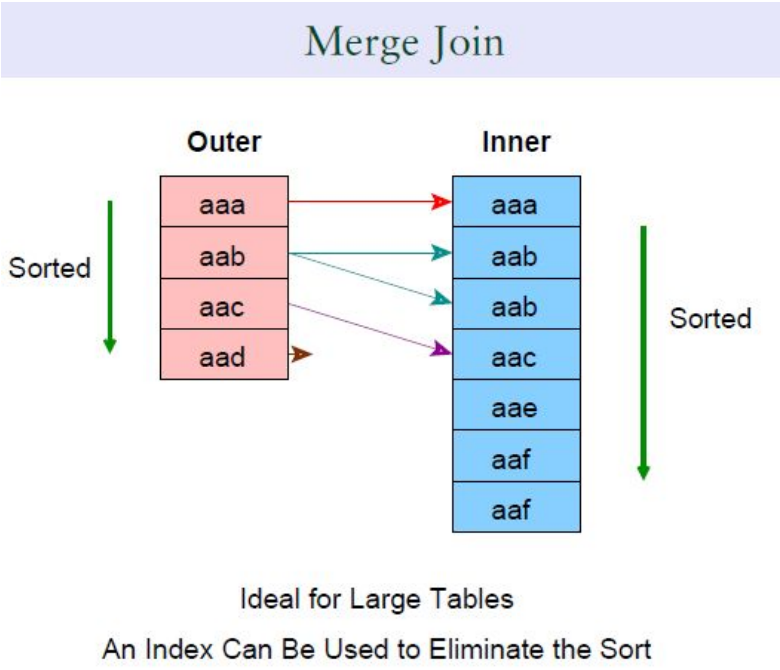


Ένα παράδειγμα για την επιλογή μεθόδου σύνδεσης στην Postgres

- Σύνδεση των δύο tables χωρίς συνθήκη επιλογής

```
19  EXPLAIN SELECT sample2.junk
20  FROM sample1 JOIN sample2 ON (sample1.id = sample2.id)
```

	QUERY PLAN text
1	Merge Join (cost=1526.80..2837.33 rows=87012 width=32)
2	Merge Cond: (sample2.id = sample1.id)
3	-> Sort (cost=74.54..77.21 rows=1070 width=48)
4	Sort Key: sample2.id
5	-> Seq Scan on sample2 (cost=0.00..20.70 rows=1070 width=48)
6	-> Sort (cost=1452.26..1492.92 rows=16264 width=16)
7	Sort Key: sample1.id
8	-> Seq Scan on sample1 (cost=0.00..314.64 rows=16264 width=1...



Επιλογή merge join για φόρτους μεγάλου μεγέθους

Ένα παράδειγμα για την επιλογή μεθόδου σύνδεσης στην Postgres

- Σύνδεση των δύο tables χωρίς συνθήκη επιλογής. Η σειρά της σύνδεσης των δύο tables δεν είναι σημαντική.

```
22  EXPLAIN SELECT sample2.junk
23  FROM sample2 JOIN sample1 ON (sample2.id = sample1.id)
```

	QUERY PLAN text
1	Merge Join (cost=1526.80..2837.33 rows=87012 width=32)
2	Merge Cond: (sample2.id = sample1.id)
3	-> Sort (cost=74.54..77.21 rows=1070 width=48)
4	Sort Key: sample2.id
5	-> Seq Scan on sample2 (cost=0.00..20.70 rows=1070 width=48)
6	-> Sort (cost=1452.26..1492.92 rows=16264 width=16)
7	Sort Key: sample1.id
8	-> Seq Scan on sample1 (cost=0.00..314.64 rows=16264 width=1...

Το μικρότερο sample2 table είναι πάντα στην εξωτερική πλευρά του merge join

Ένα παράδειγμα για την επιλογή μεθόδου σύνδεσης στην Postgres

- Δημιουργία στατιστικών

```
25  ANALYSE sample1;
26
27  ANALYSE sample2;
28
29  EXPLAIN SELECT sample2.junk
30  FROM sample1 JOIN sample2 ON (sample1.id = sample2.id)
```

	QUERY PLAN text
1	Hash Join (cost=14.38..263.25 rows=4079 width=254)
2	Hash Cond: (sample1.id = sample2.id)
3	-> Seq Scan on sample1 (cost=0.00..192.79 rows=4079 width=4)
4	-> Hash (cost=11.39..11.39 rows=239 width=258)
5	-> Seq Scan on sample2 (cost=0.00..11.39 rows=239 width=258)

Ένα παράδειγμα για την επιλογή μεθόδου σύνδεσης στην Postgres

- Τα εξωτερικά joins μπορεί να επηρεάσουν την επιλογή των συνδέσεων από το βελτιστοποιητή

```
32  EXPLAIN SELECT sample1.junk
33  FROM sample1 RIGHT OUTER JOIN sample2 ON (sample1.id = sample2.id);
```

	QUERY PLAN text
1	Hash Right Join (cost=14.38..263.25 rows=4079 width=254)
2	Hash Cond: (sample1.id = sample2.id)
3	-> Seq Scan on sample1 (cost=0.00..192.79 rows=4079 width=258)
4	-> Hash (cost=11.39..11.39 rows=239 width=4)
5	-> Seq Scan on sample2 (cost=0.00..11.39 rows=239 width=4)

Ένα παράδειγμα για την επιλογή μεθόδου σύνδεσης στην Postgres

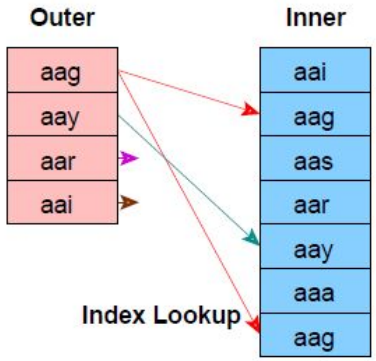
- Δημιουργία ευρετηρίων

```
35  CREATE INDEX i_sample1 on sample1 (id);
36  CREATE INDEX i_sample2 on sample2 (id);
42  EXPLAIN SELECT sample2.junk
43  FROM sample1 JOIN sample2 ON (sample1.id = sample2.id)
44  WHERE sample1.id > 'BOL' and sample1.id < 'BRA'
```

	QUERY PLAN text
1	Nested Loop (cost=0.43..16.47 rows=1 width=254)
2	-> Index Only Scan using i_sample1 on sample1 (cost=0.28..8.30 rows=1 widt
3	Index Cond: ((id > 'BOL'::bpchar) AND (id < 'BRA'::bpchar))
4	-> Index Scan using i_sample2 on sample2 (cost=0.14..8.16 rows=1 width=25
5	Index Cond: (id = sample1.id)

Ήταν ~436 με hash join

Nested Loop Join with Inner Index Scan



No Setup Required

Index Must Already Exist

Ένα παράδειγμα για την επιλογή μεθόδου σύνδεσης στην Postgres

- Η επιλογή των συνθηκών όπως και η χρήση του LIMIT επηρεάζουν την επιλογή του join.

```
46  EXPLAIN SELECT sample2.junk
47  FROM sample1 JOIN sample2 ON (sample1.id = sample2.id)
48  WHERE sample2.junk ~ '^xxx';
```

	QUERY PLAN text
1	Hash Join (cost=14.98..263.85 rows=4079 width=254)
2	Hash Cond: (sample1.id = sample2.id)
3	-> Seq Scan on sample1 (cost=0.00..192.79 rows=4079 width=4)
4	-> Hash (cost=11.99..11.99 rows=239 width=258)
5	-> Seq Scan on sample2 (cost=0.00..11.99 rows=239 width=258)
6	Filter: (junk ~ '^xxx'::text)

Επιλέχθηκε hash join γιατί αναμένονται πολλά tuples. Ο μικρότερος πίνακας sample2 πάντα γίνεται hashed.

Ένα παράδειγμα για την επιλογή μεθόδου σύνδεσης στην Postgres

- Χωρίς LIMIT επιλέγεται hash join για συνδέσεις χωρίς συνθήκη επιλογής

```
50  EXPLAIN SELECT sample2.junk
51  FROM sample1 JOIN sample2 ON (sample1.id = sample2.id)
52  ORDER BY 1;
```

	QUERY PLAN text
1	Sort (cost=507.87..518.07 rows=4079 width=254)
2	Sort Key: sample2.junk
3	-> Hash Join (cost=14.38..263.25 rows=4079 width=254)
4	Hash Cond: (sample1.id = sample2.id)
5	-> Seq Scan on sample1 (cost=0.00..192.79 rows=4079 width=4)
6	-> Hash (cost=11.39..11.39 rows=239 width=258)
7	-> Seq Scan on sample2 (cost=0.00..11.39 rows=239 width=258)

Ένα παράδειγμα για την επιλογή μεθόδου σύνδεσης στην Postgres

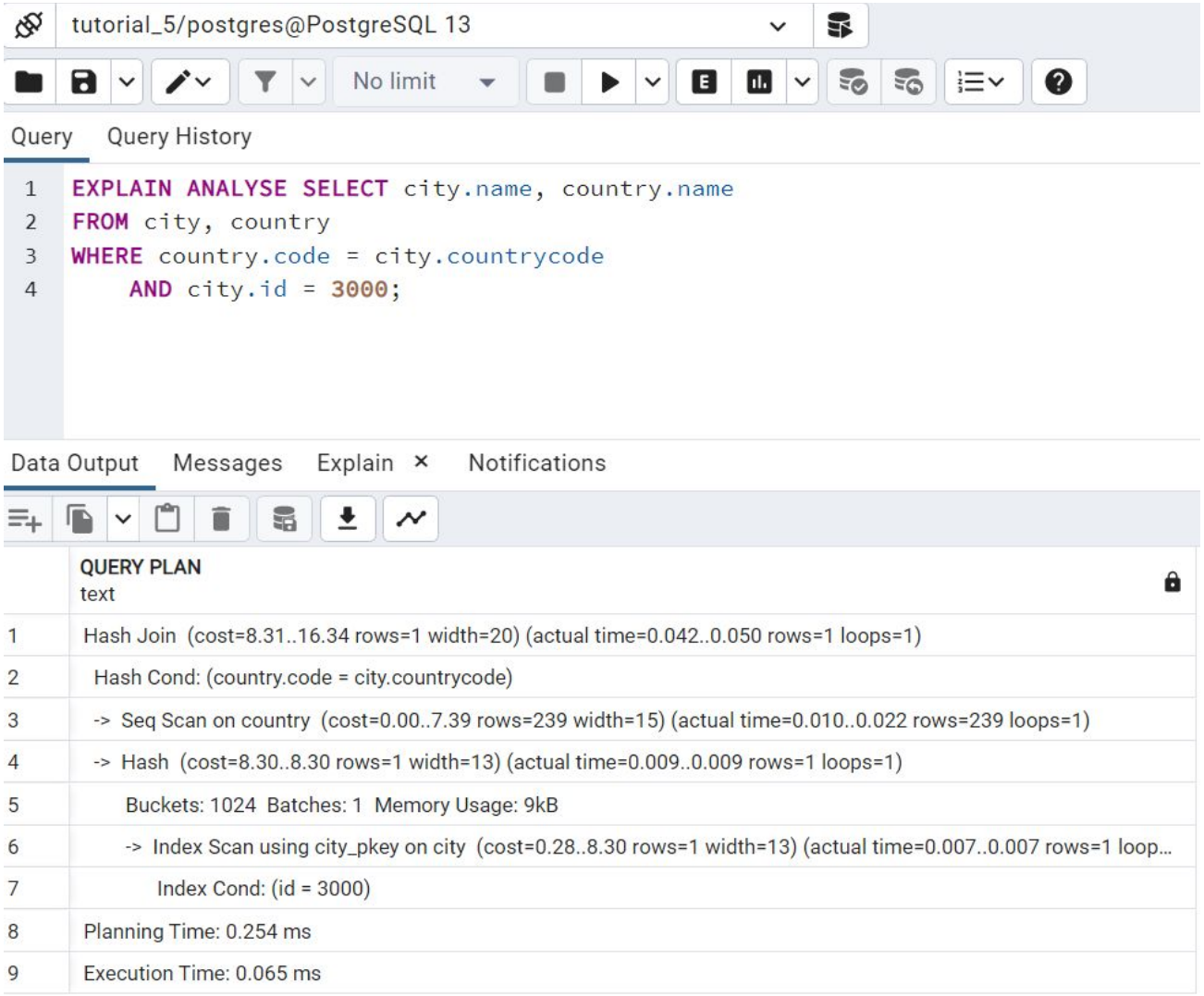
- Με LIMIT επιλέγεται merge join (μπορεί και nested loop join ανάλογα με το πλήθος) για συνδέσεις χωρίς συνθήκη επιλογής

```
54  EXPLAIN SELECT sample2.id, sample2.junk
55  FROM sample1 JOIN sample2 ON (sample1.id = sample2.id)
56  ORDER BY 1
57  LIMIT 100;
```

	QUERY PLAN text
1	Limit (cost=0.43..19.90 rows=100 width=258)
2	-> Merge Join (cost=0.43..794.62 rows=4079 width=258)
3	Merge Cond: (sample2.id = sample1.id)
4	-> Index Scan using i_sample2 on sample2 (cost=0.14..47.73 rows=239 width=258)
5	-> Index Only Scan using i_sample1 on sample1 (cost=0.28..695.31 rows=4079 width=4)

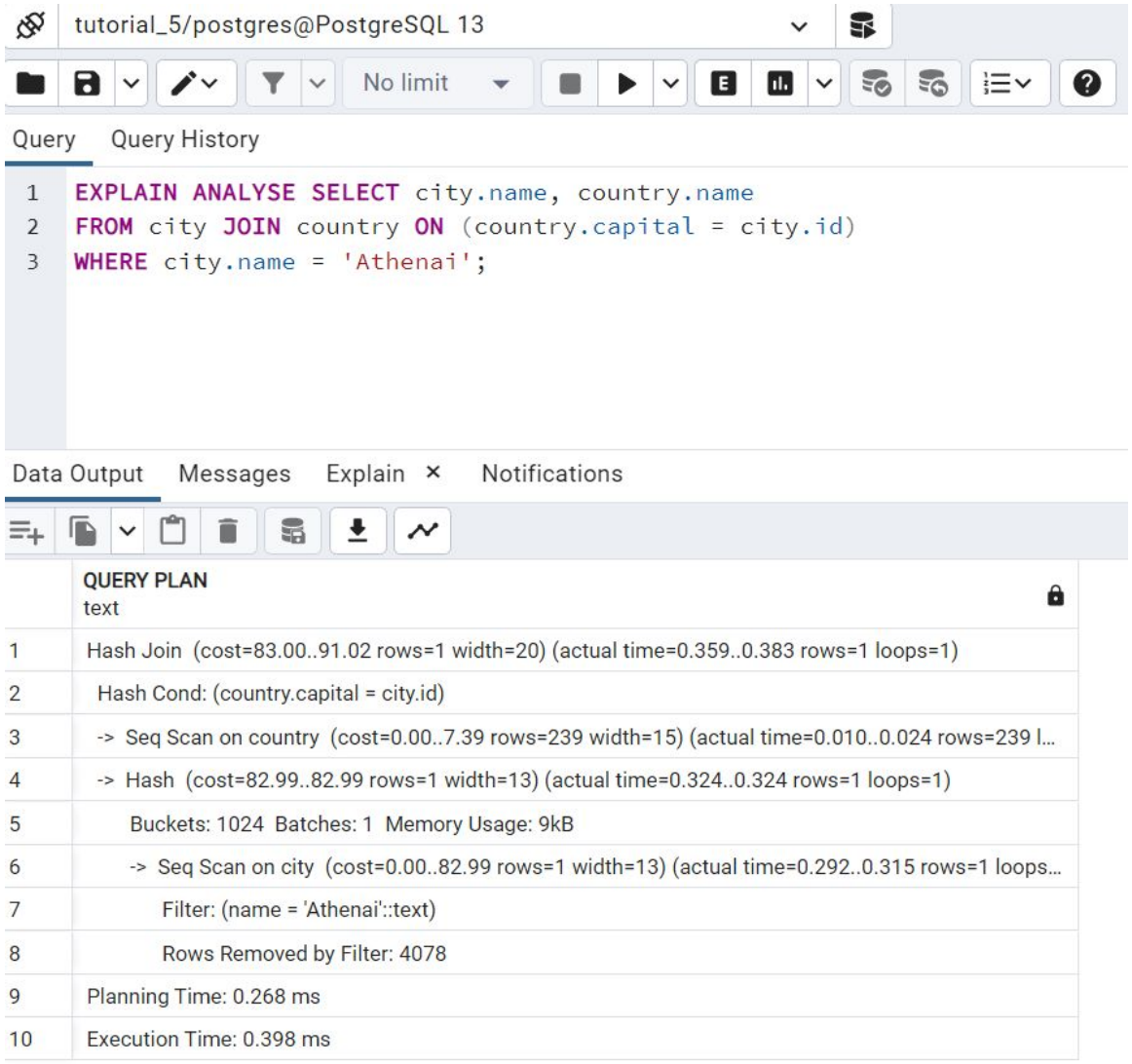
Βελτίωση απόδοσης συνδέσεων (1/3)

- Τα κύρια κλειδιά έχουν ευρετήρια



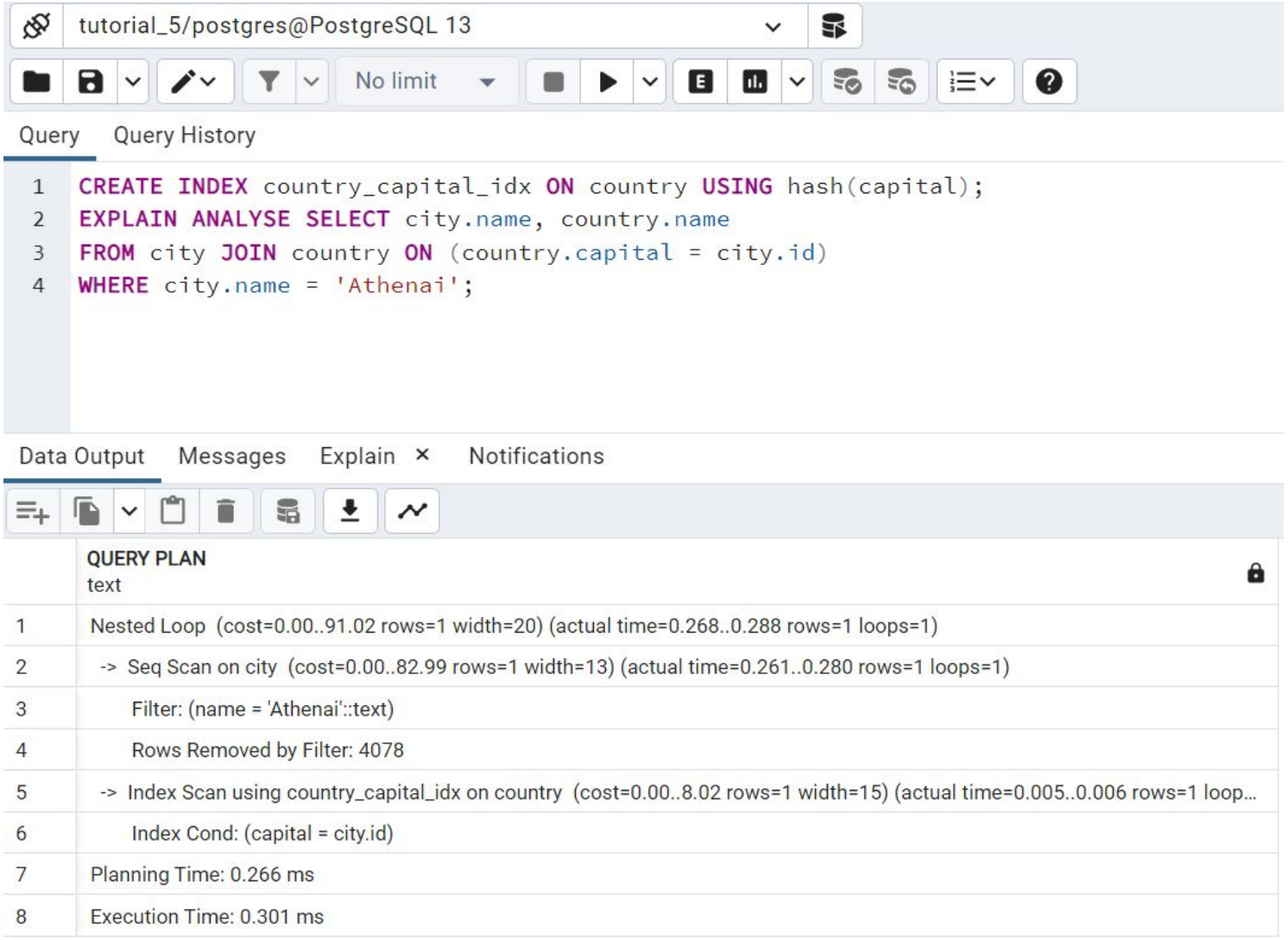
Βελτίωση απόδοσης συνδέσεων (2/3)

- Τα ξένα κλειδιά δεν έχουν ευρετήρια.



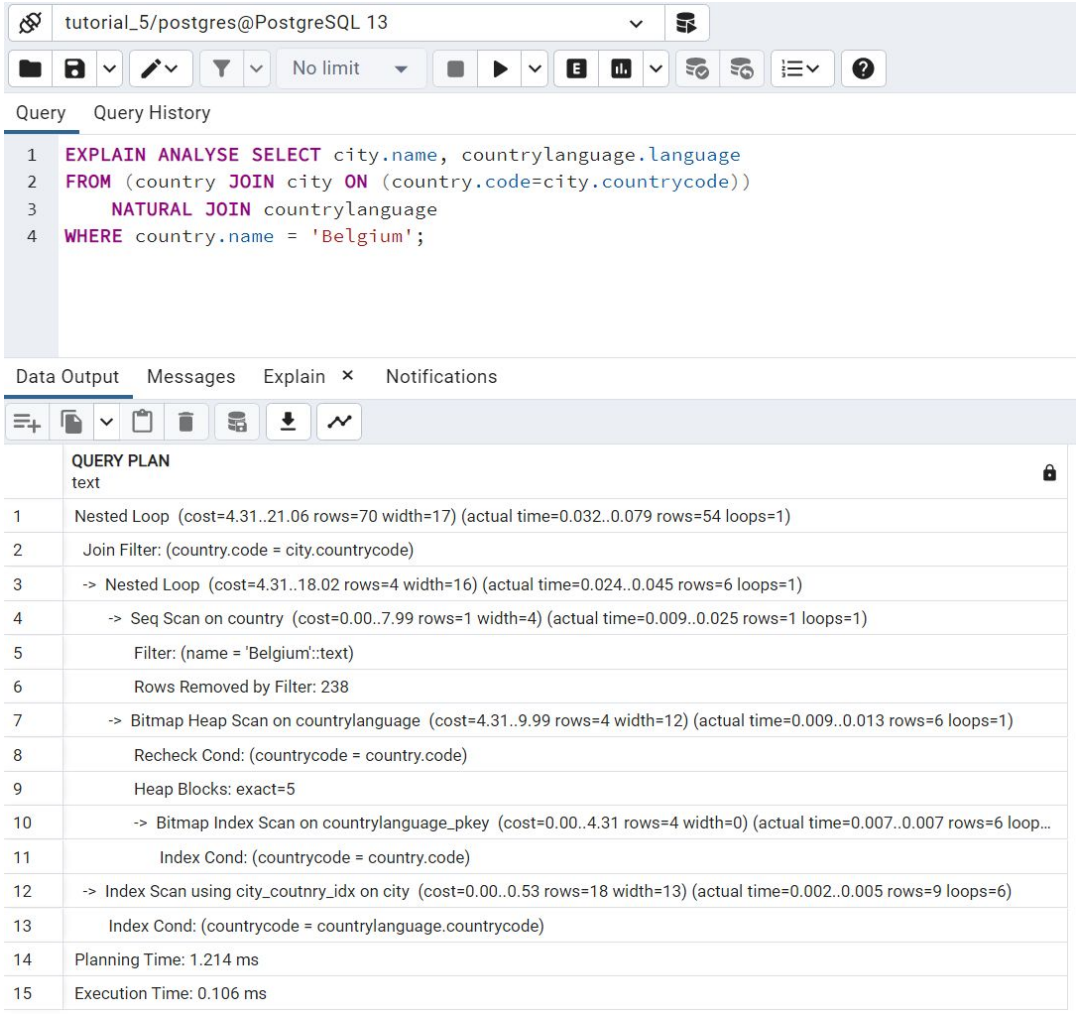
Βελτίωση απόδοσης συνδέσεων (3/3)

- Δημιουργία νέου ευρετηρίου για το ξένο κλειδί...



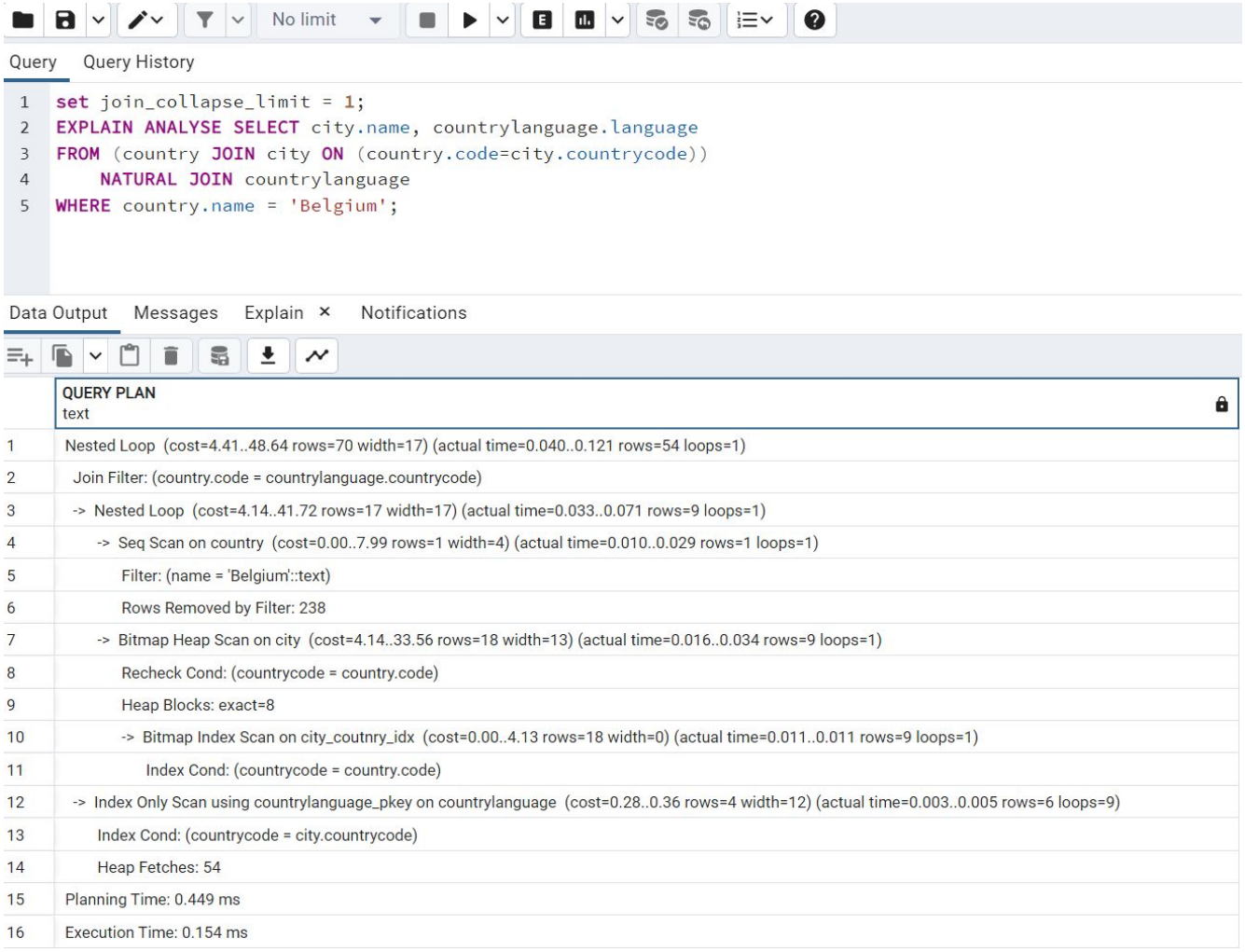
Ελέγχοντας τη σειρά των συνδέσεων (1/2)

- Η σειρά των συνδέσεων επηρεάζει την απόδοση του συστήματος. Τη σειρά την αποφασίζει ο βελτιστοποιητής

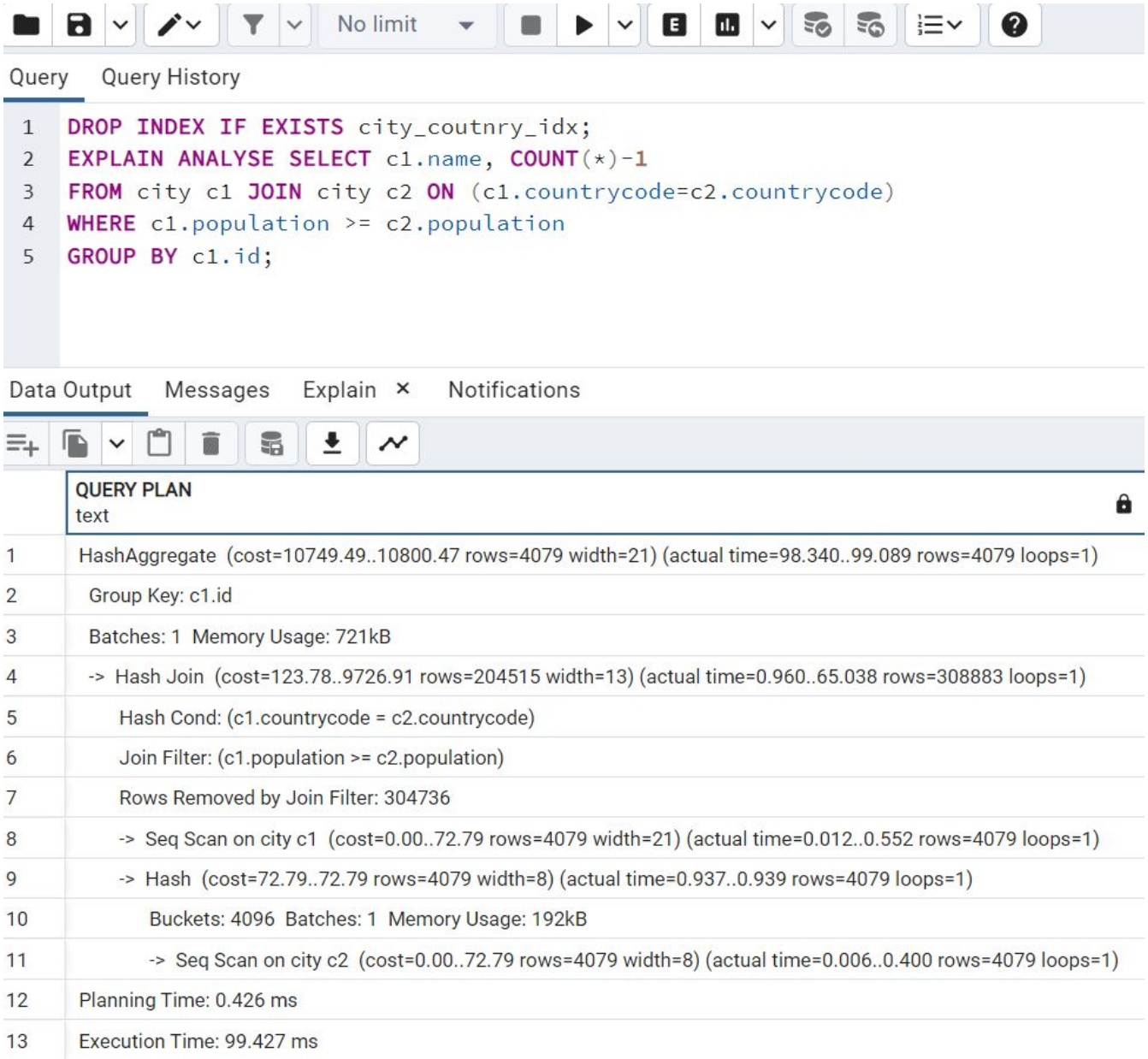


Ελέγχοντας τη σειρά των συνδέσεων (2/2)

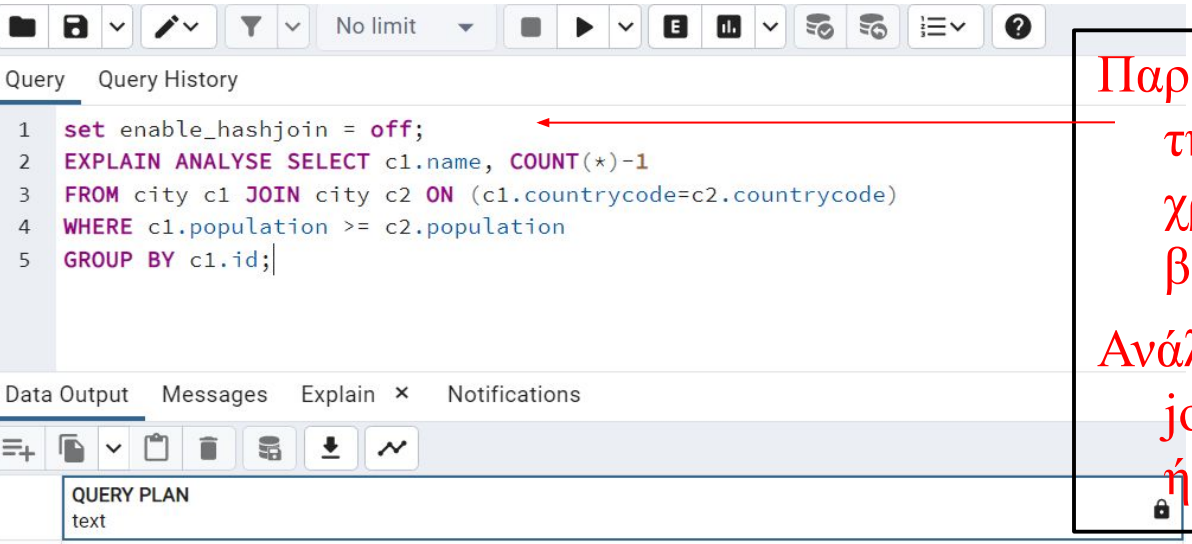
- Πράξεις σύνδεσης: προσπέλαση πινάκων (FROM) και συνθήκες (WHERE)
- Σε μια συνεδρία (session) μπορούμε να επιβάλουμε τη σειρά συνδέσεων που εμφανίζονται στο FROM θέτοντας join_collapse_limit=1



Αλλάζοντας τρόπο υπολ. συνδέσεων (1/3)



Αλλάζοντας τρόπο υπολ. συνδέσεων (2/3)



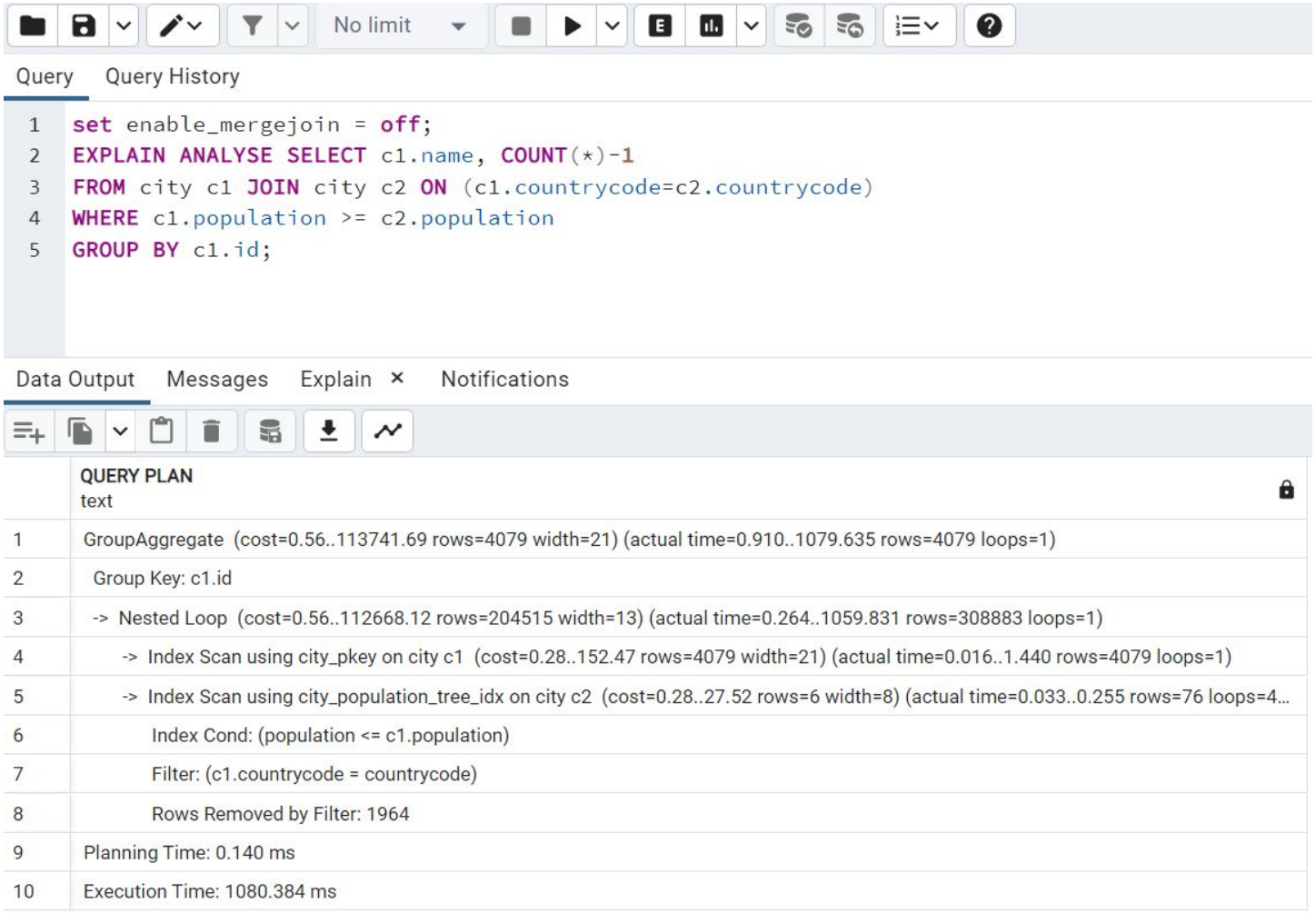
	QUERY PLAN text
1	HashAggregate (cost=12414.81..12465.79 rows=4079 width=21) (actual time=125.882..126.388 rows=4079 loops=1)
2	Group Key: c1.id
3	Batches: 1 Memory Usage: 721kB
4	-> Merge Join (cost=634.82..11392.23 rows=204515 width=13) (actual time=7.611..94.019 rows=308883 loops=1)
5	Merge Cond: (c1.countrycode = c2.countrycode)
6	Join Filter: (c1.population >= c2.population)
7	Rows Removed by Join Filter: 304736
8	-> Sort (cost=317.41..327.61 rows=4079 width=21) (actual time=3.778..4.488 rows=4079 loops=1)
9	Sort Key: c1.countrycode
10	Sort Method: quicksort Memory: 414kB
11	-> Seq Scan on city c1 (cost=0.00..72.79 rows=4079 width=21) (actual time=0.011..0.418 rows=4079 loops=1)
12	-> Sort (cost=317.41..327.61 rows=4079 width=8) (actual time=3.827..21.890 rows=613614 loops=1)
13	Sort Key: c2.countrycode
14	Sort Method: quicksort Memory: 288kB
15	-> Seq Scan on city c2 (cost=0.00..72.79 rows=4079 width=8) (actual time=0.012..0.521 rows=4079 loops=1)
16	Planning Time: 0.405 ms
17	Execution Time: 126.806 ms

Παράμετροι που ελέγχουν τα είδη της σύνδεσης που μπορεί να χρησιμοποιεί ο βελτιστοποιητής.

Ανάλογη χρήση με το join_collapse_limit που είδαμε ήδη

Αλλάζοντας τρόπο υπολ. συνδέσεων (3/3)

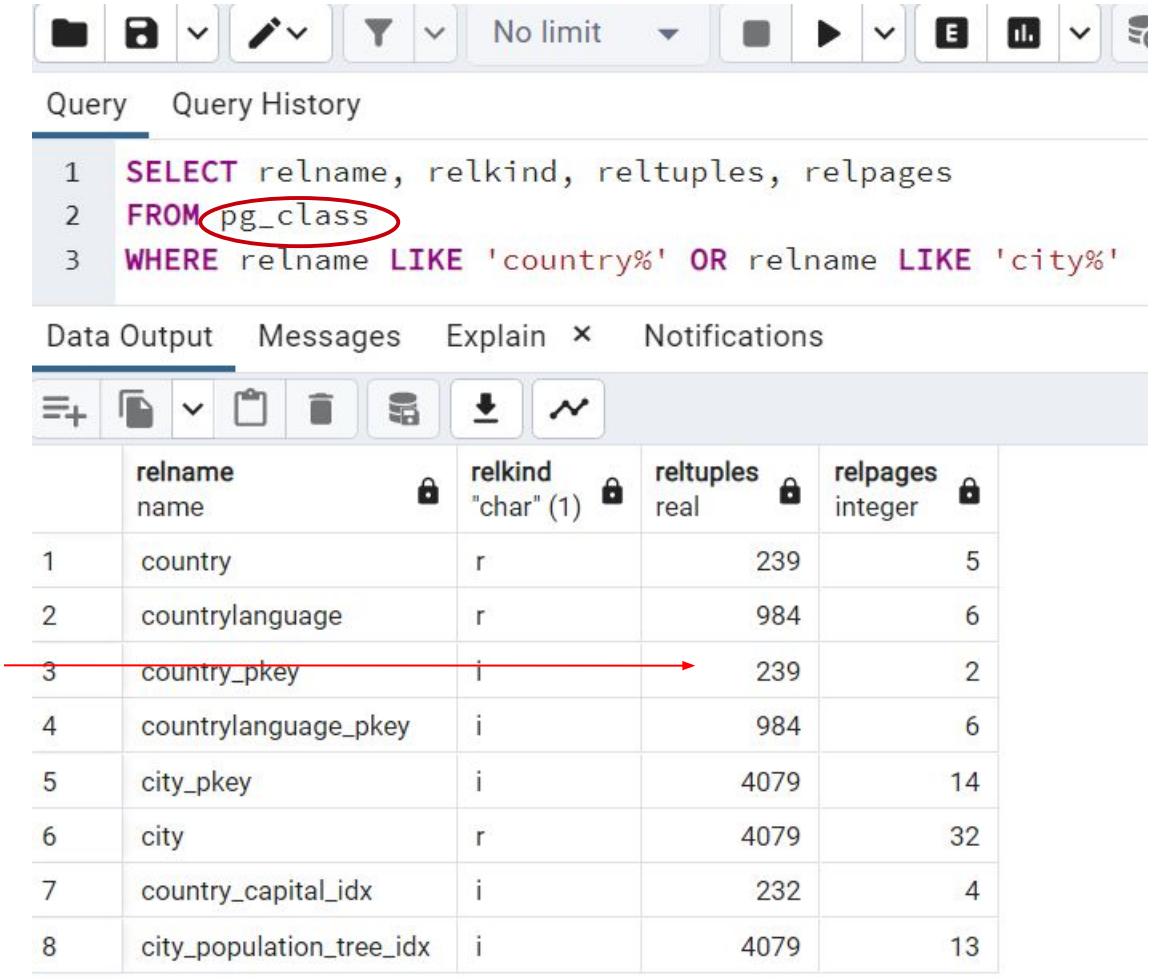
- Ας δούμε τι συμβαίνει όταν, μετά την απενεργοποίηση των hash joins απενεργοποιήσουμε και τα merge joins.
- Τι παρατηρείται σχετικά με την απόδοση των τριών αυτών σχεδίων;



Χρήση στατιστικών

Ο Κατάλογος κρατά στατιστική πληροφορία την οποία αξιοποιεί ο βελτιστοποιητής για να εκτιμήσει το κόστος κάθε σχεδίου υπολογισμού ενός αιτήματος.

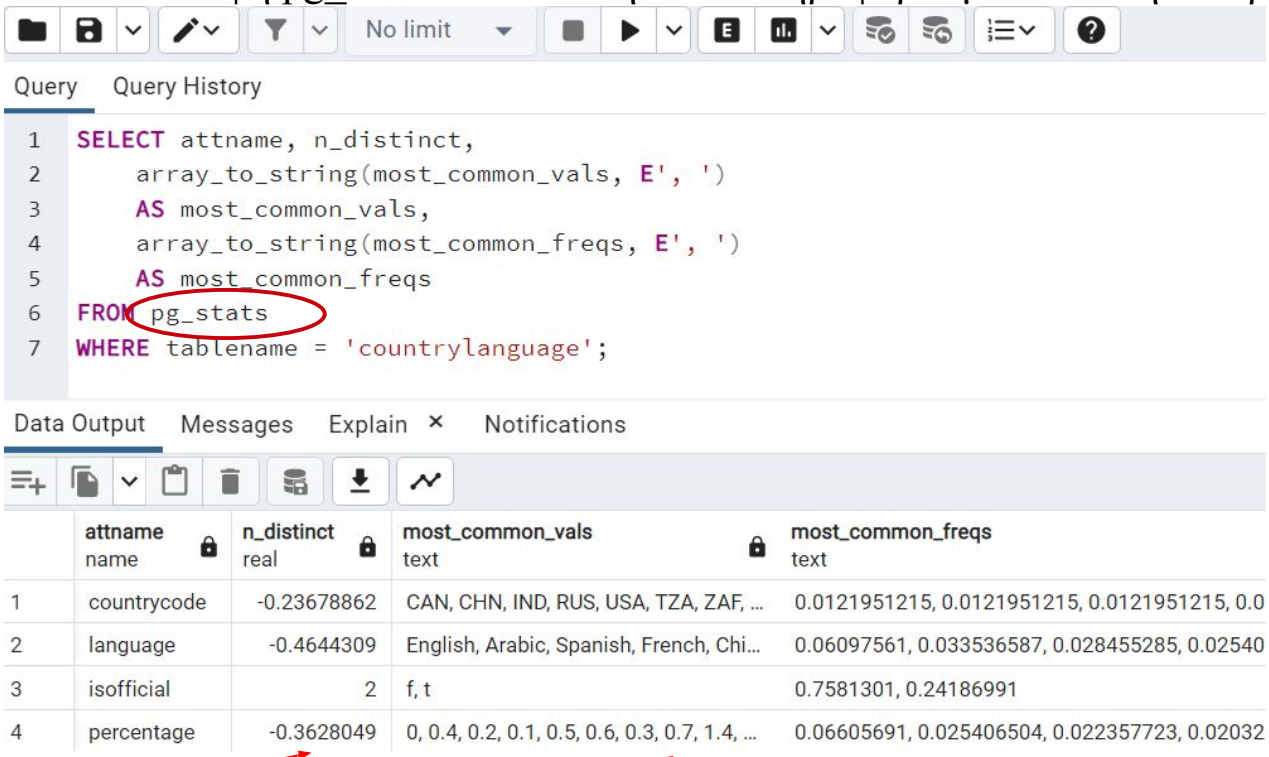
Ο πίνακας καταλόγου pg_class περιέχει πληροφορίες για το πλήθος των πλειάδων και των σελίδων στο δίσκο για κάθε σχέση/πίνακα



	relname name	relkind "char" (1)	reltuples real	relpages integer
1	country	r	239	5
2	countrylanguage	r	984	6
3	country_pkey	i	239	2
4	countrylanguage_pkey	i	984	6
5	city_pkey	i	4079	14
6	city	r	4079	32
7	country_capital_idx	i	232	4
8	city_population_tree_idx	i	4079	13

Στατιστικά χρήσιμα για εκτίμηση επιλεκτικότητας (1/2)

Στον πίνακα καταλόγου pg_statistic υπάρχουν προσεγγιστικά στατιστικά για τις πιο συχνές τιμές σε κάθε πεδίο κάθε πίνακα. Η όψη pg_stats που δίνει την ίδια πληροφορία με πιο εύληπτο τρόπο.



Πεδίο n_distinct:

- **>0 :** εκτιμώμενο πλήθος διακριτών τιμών πεδίου
- **<0 :** (αρνητικός) λόγος του πλήθους των τιμών του πεδίου προς το πλήθος των εγγραφών του πίνακα

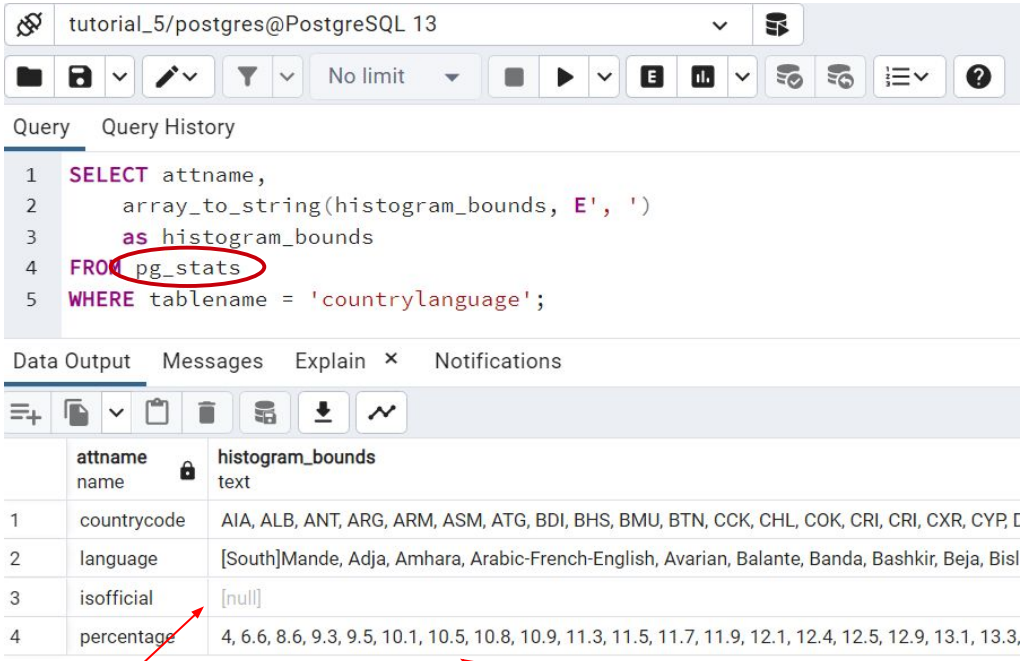
Η όψη καταλόγου pg_stats περιέχει πληροφορίες για τις πιο συχνές τιμές κάθε πεδίου

Στατιστικά χρήσιμα για εκτίμηση επιλεκτικότητας (2/2)

Στον πίνακα καταλόγου pg_statistic υπάρχουν και τα προσεγγιστικά ιστογράμματα τιμών για κάθε πεδίο κάθε πίνακα. Η όψη pg_stats δίνει και αυτή την πληροφορία με πιο εύληπτο τρόπο.

Η λίστα των τιμών του ιστογράμματος διαμερίζουν το σύνολο των τιμών του πεδίου σε ομάδες με ίδιο πληθυσμό τιμών. Εφόσον υπάρχουν και συχνές τιμές (most_common_vals), δεν λαμβάνονται υπόψη στον υπολογισμό του ιστογράμματος.

Παίρνει την τιμή NULL αν ο τύπος του πεδίου δεν υποστηρίζει τελεστή < καθώς και αν οι συχνές τιμές έχουν ήδη καλύψει όλο τον πληθυσμό τιμών.



Η όψη καταλόγου pg_stats περίεχει πληροφορίες για τα ιστογράμματα των τιμών κάθε πεδίου

Ρύθμιση ακρίβειας στατιστικών

- **default_statistics_target**
  Μεταβλητή συστήματος (configuration variable) που προσδιορίζει το μέγιστο πλήθος τιμών στα most_common_vals και histogram_bounds
- Η προκαθορισμένη τιμή αυτής της μεταβλητής είναι 100. Ο διαχειριστής μπορεί να την τροποποιήσει με την εντολή:
  SET default_statistics_target = 10;
- **ALTER TABLE SET STATISTICS**

Επιτρέπει στο διαχειριστή την αλλαγή του ορίου για συγκεκριμένο πίνακα/πεδίο.

- Έχει αποδειχθεί ότι η χρήση πολύ αναλυτικών στατιστικών δεν βελτιώνει την εκτίμηση κόστους καθώς οι εκτιμήσεις εμπεριέχουν σφάλματα. Πώς μπορείτε να αξιοποιήσετε αυτό το γεγονός για να αποφύγετε την τήρηση περιττών στατιστικών;

Ενημέρωση στατιστικών

Η ενημέρωση των στατιστικών επηρεάζει το παραγόμενο σχέδιο εκτέλεσης από το βελτιστοποιητή

- Για την ενημέρωση των στατιστικών όλης της βάσης:

VACUUM ANALYSE

- Ενημέρωση στατιστικών για συγκεκριμένο πίνακα:

VACUUM ANALYSE city

- Εκτός από την ενημέρωση των στατιστικών γίνεται και συλλογή απορριμμάτων (garbage collection). Για παράδειγμα αποδεσμεύεται χώρος που αφορά πλειάδες που έχουν διαγραφεί.

Σημαντικά κεφάλαια εγχειριδίου αναφοράς PostgreSQL (version 13)

- **Κεφ. 14 – Performance Tips**
  Αναλυτική περιγραφή χρήσης του explain και του explain analyse, γενική αναφορά στη χρήση στατιστικών από το βελτιστοποιητή, έλεγχος σειράς συνδέσεων (joins), φόρτωση μεγάλου όγκου δεδομένων.
- **Ενότητα 19.7 – Query Planning**

Περιγράφει όλες τις παραμέτρους (configuration parameters) που μπορεί να μεταβάλει ο διαχειριστής μιας βάσης για να αλλάξει το υπολογιζόμενο βέλτιστο σχέδιο υπολογισμού.

- **Κεφάλαιο 71 – How the Planner Uses Statistics**

Περιγράφει πώς από τα στατιστικά υπολογίζεται η εκτίμηση του κόστους υπολογισμού ερωτήσεων (π.χ. Πώς γίνεται η εκτίμηση της επιλεκτικότητας μια συνθήκης, του μεγέθους του αποτελέσματος μια ερώτησης κ.λ.π.)