PostgreSQL Sorgu Analizleri

Erdinc Akkaya
erdinc.akkaya@markafoni.com
@hzroot (twitter,irc,mail,im vs..)

kısa kısa..

- sorunlu sorguları bulalım.
- extensions ve diğer araçlar
- explain nedir? ne anlatır?
- vacuum ve sorgumuza olan etkileri
- index scan,sequential scan, bitmap index scan
- neden index kullanmıyor bu sorgu?
- sorguları yeniden yazalım

pg_stat_statements

- *pg_stat_statements*
  - *postgresql.conf dosyasında shared_preload_libreries kısmına eklenmeli*
  - *Ekstra bellek istediği için restart gerektirir.*
  - *Kaç block okundu,yazıldı gibi daha detaylı bilgi verir.*
  - *Sql cümlecikleriyle detayları göreceğimizden başka sistem `view`larıyla birleştirilip detaylı analizler yapmamızı sağlar.*
  - *http://www.postgresql.org/docs/current/static/pgstatstatements.html*

```
userid            | 10
dbid              | 16488
query             | UPDATE pgbench_accounts SET abalance = abalance + -774 WHERE aid = 9756913;
calls             | 1
total_time        | 0.196066
rows              | 1
shared_blks_hit   | 6
shared_blks_read  | 11
shared_blks_written | 0
local_blks_hit    | 0
local_blks_read   | 0
local_blks_written | 0
temp_blks_read    | 0
temp_blks_written | 0
-[ RECORD 4 ]
userid            | 10
dbid              | 16488
query             | SELECT abalance FROM pgbench_accounts WHERE aid = 14436125;
calls             | 1
total_time        | 4.3e-05
rows              | 1
shared_blks_hit   | 6
shared_blks_read  | 0
shared_blks_written | 0
local_blks_hit    | 0
local_blks_read   | 0
local_blks_written | 0
temp_blks_read    | 0
temp_blks_written | 0
```

pgFouine

- *PgFouine*
  - *pg_log.*
  - *log_min_statemets*
  - *php.*
  - *Restart gerekmez*
  - *pgfouine.projects.postgresql.org*

pgFouine: PostgreSQL log analysis report

Overall statistics | Queries by type | Slowest queries | Queries that took up the most time (N) | Most frequent queries (N) | Slowest queries (N)

Normalized reports are marked with a "(N)".

- Generated on 2007-03-30 20:58
- Parsed /home/gsmet/work/pgfouine/logs_pgbench_20051211063633.log (608,204 lines) in 5m16s
- Log from 2006-12-08 11:14:50 to 2006-12-11 06:36:31
- Executed on edgewood

Overall statistics

- Number of unique normalized queries: 9
- Number of queries: 257,288
- Total query duration: 1951h35m33s
- First query: 2006-12-08 11:14:50
- Last query: 2006-12-11 06:36:18
- Query peak: 61 queries/s at 2006-12-08 11:14:50

Queries by type

Type	Count	Percentage
SELECT	210,483	81.8
UPDATE	46,774	18.2

Slowest queries

Rank	Duration (s)	Query
1	4,777.68	UPDATE accounts SET filler=lower('teSt fiLlEr') WHERE aid < 1000;
2	3,949.61	UPDATE accounts SET filler=lower('teSt fiLlEr') WHERE aid < 1000;
3	3,763.44	UPDATE accounts SET filler=lower('teSt fiLlEr') WHERE aid < 1000;
4	3,696.27	UPDATE accounts SET filler=lower('teSt fiLlEr') WHERE aid < 1000;
5	3,687.32	UPDATE accounts SET filler=lower('teSt fiLlEr') WHERE aid < 1000;
6	3,672.68	UPDATE accounts SET filler=lower('teSt fiLlEr') WHERE aid < 1000;
7	3,539.89	UPDATE accounts SET filler=lower('teSt fiLlEr') WHERE aid < 1000;
8	3,418.57	UPDATE accounts SET filler=lower('teSt fiLlEr') WHERE aid < 1000;

EXPLAIN

- Sorguyu yürütmez
- `Planner`
- Zaman yok
- Tahmini

```
dellstore2=# EXPLAIN SELECT * FROM orderlines ol
left join orders o on o.orderid = ol.orderid
left join products p on p.prod_id = ol.prod_id
left join customers c on c.customerid = o.customerid
left join cust_hist ch on ch.orderid = o.orderid and ch.customerid = c.customerid
order by p.prod_id;

                                                                    QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------
 Sort  (cost=73748.09..73898.97 rows=60350 width=1918)
   Sort Key: p.prod_id
   ->  Hash Left Join  (cost=6048.61..18828.98 rows=60350 width=1918)
         Hash Cond: (ol.prod_id = p.prod_id)
         ->  Merge Right Join  (cost=5722.61..16994.23 rows=60350 width=1652)
               Merge Cond: (o.orderid = ol.orderid)
               ->  Merge Left Join  (cost=5722.61..13096.35 rows=12000 width=1634)
                     Merge Cond: (o.orderid = ch.orderid)
                     Join Filter: (ch.customerid = c.customerid)
                     ->  Nested Loop Left Join  (cost=0.00..6287.61 rows=12000 width=1622)
                           ->  Index Scan using orders_pkey on orders o  (cost=0.00..720.25 rows=12000 width=60)
                           ->  Index Scan using customers_pkey on customers c  (cost=0.00..0.45 rows=1 width=1562)
                                 Index Cond: (customerid = o.customerid)
                     ->  Sort  (cost=5722.61..5873.49 rows=60350 width=12)
                           Sort Key: ch.orderid
                           ->  Seq Scan on cust_hist ch  (cost=0.00..930.50 rows=60350 width=12)
               ->  Index Scan using ix_orderlines_orderid on orderlines ol  (cost=0.00..3113.51 rows=60350 width=18)
         ->  Hash  (cost=201.00..201.00 rows=10000 width=266)
               ->  Seq Scan on products p  (cost=0.00..201.00 rows=10000 width=266)
```

EXPLAIN ANALYZE

- Sorguyu yürütür!
- Tahmin + Gerçek
- Süre

```
dellstore2=# EXPLAIN ANALYZE SELECT  * FROM orderlines ol
left join orders o on o.orderid = ol.orderid
left join products p on p.prod_id = ol.prod_id
left join customers c on c.customerid = o.customerid
left join cust_hist ch on ch.orderid = o.orderid and ch.customerid = c.customerid
order by p.prod_id;
                                                    QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------------------------------------
 Sort  (cost=12747.89..12898.77 rows=60350 width=377) (actual time=892.208..925.751 rows=383270 loops=1)
   Sort Key: p.prod_id
             ....
             ....
             ....
   Total runtime: 1088.277 ms
```

EXPLAIN ANALYZE

```
dellstore2=# EXPLAIN ANALYZE SELECT  * FROM orderlines ol
left join orders o on o.orderid = ol.orderid
left join products p on p.prod_id = ol.prod_id
left join customers c on c.customerid = o.customerid
left join cust_hist ch on ch.orderid = o.orderid and ch.customerid = c.customerid
order by p.prod_id;
                                                                    QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 Sort  (cost=73748.09..73898.97 rows=60350 width=1918) (actual time=1189.306..1325.300 rows=383270 loops=1)
   Sort Key: p.prod_id
   Sort Method: external merge  Disk: 112512kB
   ->  Hash Left Join  (cost=6048.61..18828.98 rows=60350 width=1918) (actual time=46.441..476.380 rows=383270 loops=1)
         Hash Cond: (ol.prod_id = p.prod_id)
         ->  Merge Right Join  (cost=5722.61..16994.23 rows=60350 width=1652) (actual time=44.364..284.230 rows=383270 loops=1)
               Merge Cond: (o.orderid = ol.orderid)
               ->  Merge Left Join  (cost=5722.61..13096.35 rows=12000 width=1634) (actual time=44.348..120.306 rows=60350 loops=1)
                     Merge Cond: (o.orderid = ch.orderid)
                     Join Filter: (ch.customerid = c.customerid)
                     ->  Nested Loop Left Join  (cost=0.00..6287.61 rows=12000 width=1622) (actual time=0.056..42.540 rows=12000 loops=1)
                           ->  Index Scan using orders_pkey on orders o  (cost=0.00..720.25 rows=12000 width=60) (actual time=0.032..2.717 rows=12000 loops=1)
                           ->  Index Scan using customers_pkey on customers c  (cost=0.00..0.45 rows=1 width=1562) (actual time=0.002..0.002 rows=1 loops=12000)
                                 Index Cond: (customerid = o.customerid)
                     ->  Sort  (cost=5722.61..5873.49 rows=60350 width=12) (actual time=44.282..56.236 rows=60350 loops=1)
                           Sort Key: ch.orderid
                           Sort Method: external sort  Disk: 1536kB
                           ->  Seq Scan on cust_hist ch  (cost=0.00..930.50 rows=60350 width=12) (actual time=0.014..7.140 rows=60350 loops=1)
               ->  Index Scan using ix_orderlines_orderid on orderlines ol  (cost=0.00..3113.51 rows=60350 width=18) (actual time=0.013..64.523 rows=383262 loops=1)
         ->  Hash  (cost=201.00..201.00 rows=10000 width=266) (actual time=2.062..2.062 rows=10000 loops=1)
               Buckets: 1024  Batches: 1  Memory Usage: 814kB
               ->  Seq Scan on products p  (cost=0.00..201.00 rows=10000 width=266) (actual time=0.003..0.742 rows=10000 loops=1)
 Total runtime: 1484.790 ms
(23 rows)
```

EXPLAIN ANALYZE rows

```
dellstore2=# EXPLAIN ANALYZE SELECT  * FROM orderlines ol
left join orders o on o.orderid = ol.orderid
left join products p on p.prod_id = ol.prod_id
left join customers c on c.customerid = o.customerid
left join cust_hist ch on ch.orderid = o.orderid and ch.customerid = c.customerid
order by p.prod_id;
                                                                    QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------
 Sort  (cost=12747.89..12898.77 rows=60350 width=377) (actual time=892.208..925.751 rows=383270 loops=1)
   Sort Key: p.prod_id
   Sort Method: quicksort  Memory: 209912kB
   ->  Hash Left Join  (cost=4251.53..7955.78 rows=60350 bwidth=377) (actual time=114.796..450.834 rows=383270 loops=1)
         Hash Cond: (ol.prod_id = p.prod_id)
         ->  Hash Left Join  (cost=3925.53..6121.03 rows=60350 width=328) (actual time=112.416..295.719 rows=383270 loops=1)
               Hash Cond: (ol.orderid = o.orderid)
               ->  Seq Scan on orderlines ol  (cost=0.00..988.50 rows=60350 idth=18) (actual time=0.015..6.230 rows=60350 loops=1)
               ->  Hash  (cost=3775.53..3775.53 rows=12000 width=310) (actual time=112.383..112.383 rows=60350 loops=1)
                     Buckets: 2048  Batches: 1  Memory Usage: 14555kB
                     ->  Hash Right Join  (cost=1638.00..3775.53 rows=12000 width=310) (actual time=33.339..74.185 rows=60350 loops=1)
                           Hash Cond: ((ch.orderid = o.orderid) AND (ch.customerid = c.customerid))
                           ->  Seq Scan on cust_hist ch  (cost=0.00..930.50 rows=60350 width=12) (actual time=0.004..5.341 rows=60350 loops=1)
                           ->  Hash  (cost=1458.00..1458.00 rows=12000 width=298) (actual time=33.320..33.320 rows=12000 loops=1)
                                 Buckets: 2048  Batches: 1  Memory Usage: 2730kB
                                 ->  Hash Left Join  (cost=938.00..1458.00 rows=12000 width=298) (actual time=15.357..25.728 rows=12000 loops=1)
                                       Hash Cond: (o.customerid = c.customerid)
                                       ->  Seq Scan on orders o  (cost=0.00..220.00 rows=12000 width=30) (actual time=0.009..1.086 rows=12000 loops=1)
                                       ->  Hash  (cost=688.00..688.00 rows=20000 width=268) (actual time=15.334..15.334 rows=20000 loops=1)
                                             Buckets: 2048  Batches: 1  Memory Usage: 3852kB
                                             ->  Seq Scan on customers c  (cost=0.00..688.00 rows=20000 width=268) (actual time=0.005..5.676 rows=20000 loops=1)
         ->  Hash  (cost=201.00..201.00 rows=10000 width=49) (actual time=2.363..2.363 rows=10000 loops=1)
               Buckets: 1024  Batches: 1  Memory Usage: 814kB
               ->  Seq Scan on products p  (cost=0.00..201.00 rows=10000 width=49) (actual time=0.004..0.825 rows=10000 loops=1)
 Total runtime: 1088.277 ms
```

EXPLAIN ANALYZE external sort

```
dellstore2=# EXPLAIN ANALYZE SELECT  * FROM orderlines ol
left join orders o on o.orderid = ol.orderid
left join products p on p.prod_id = ol.prod_id
left join customers c on c.customerid = o.customerid
left join cust_hist ch on ch.orderid = o.orderid and ch.customerid = c.customerid
order by p.prod_id;

                                                                  QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------
 Sort  (cost=73748.09..73898.97 rows=60350 width=1918) (actual time=1189.306..1325.300 rows=383270 loops=1)
   Sort Key: p.prod_id
   Sort Method: external merge  Disk: 112512kB
   ->  Hash Left Join  (cost=6048.61..18828.98 rows=60350 width=1918) (actual time=46.441..476.380 rows=383270 loops=1)
         Hash Cond: (ol.prod_id = p.prod_id)
         ->  Merge Right Join  (cost=5722.61..16994.23 rows=60350 width=1652) (actual time=44.364..284.230 rows=383270 loops=1)
               Merge Cond: (o.orderid = ol.orderid)
               ->  Merge Left Join  (cost=5722.61..13096.35 rows=12000 width=1634) (actual time=44.348..120.306 rows=60350 loops=1)
                     Merge Cond: (o.orderid = ch.orderid)
                     Join Filter: (ch.customerid = c.customerid)
                     ->  Nested Loop Left Join  (cost=0.00..6287.61 rows=12000 width=1622) (actual time=0.056..42.540 rows=12000 loops=1)
                           ->  Index Scan using orders_pkey on orders o  (cost=0.00..720.25 rows=12000 width=60) (actual time=0.032..2.717 rows=12000 loops=1)
                           ->  Index Scan using customers_pkey on customers c  (cost=0.00..0.45 rows=1 width=1562) (actual time=0.002..0.002 rows=1 loops=12000)
                                 Index Cond: (customerid = o.customerid)
                     ->  Sort  (cost=5722.61..5873.49 rows=60350 width=12) (actual time=44.282..56.236 rows=60350 loops=1)
                           Sort Key: ch.orderid
                           Sort Method: external sort  Disk: 1536kB
                           ->  Seq Scan on cust_hist ch  (cost=0.00..930.50 rows=60350 width=12) (actual time=0.014..7.140 rows=60350 loops=1)
               ->  Index Scan using ix_orderlines_orderid on orderlines ol  (cost=0.00..3113.51 rows=60350 width=18) (actual time=0.013..64.523 rows=383262 loops=1)
         ->  Hash  (cost=201.00..201.00 rows=10000 width=266) (actual time=2.062..2.062 rows=10000 loops=1)
               Buckets: 1024  Batches: 1  Memory Usage: 814kB
               ->  Seq Scan on products p  (cost=0.00..201.00 rows=10000 width=266) (actual time=0.003..0.742 rows=10000 loops=1)
 Total runtime: 1484.790 ms
(23 rows)
```

EXPLAIN ANALYZE external sort 2

```
                                    QUERY PLAN
--------------------------------------------------------------------------------------------------------------
 Sort  (cost=73748.09..73898.97 rows=60350 width=1918) (actual time=1189.306..1325.300 rows=383270 loops=1)
   Sort Key: p.prod_id
   Sort Method: external merge  Disk: 112512kB

dellstore2=# show work_mem;
 work_mem
----------
 4MB
```

- 112MB disk üzerinde sort edilmiş.
- 2 katı büyüklükte bir work_mem yeterli olacaktır.

```
dellstore2=# set work_mem to 250000;
                                    QUERY PLAN
--------------------------------------------------------------------------------------------------------------
 Sort  (cost=12747.89..12898.77 rows=60350 width=1918) (actual time=896.234..929.873 rows=383270 loops=1)
   Sort Key: p.prod_id
   Sort Method: quicksort  Memory: 209912kB
   ->  Hash Left Join  (cost=4251.53..7955.78 rows=60350 width=1918) (actual time=121.257..457.502 rows=383270 loops=1)
         Hash Cond: (ol.prod_id = p.prod_id)
                ....
              Buckets: 2048  Batches: 1  Memory Usage: 14555kB
                ....
 Total runtime: 1095.512 ms
```

Cost = ?

```
dellstore2=# EXPLAIN ANALYZE SELECT * FROM customers;
                                                    QUERY PLAN
-------------------------------------------------------------------------------------------------------------------
 Seq Scan on customers  (cost=0.00..688.00 rows=20000 width=1562) (actual time=0.015..5.588 rows=20000 loops=1)
 Total runtime: 8.030 ms

dellstore2=# select name,setting from pg_settings where name like '%cost';

        name         | setting
---------------------+---------
 cpu_index_tuple_cost | 0.005
 cpu_operator_cost    | 0.0025
 cpu_tuple_cost       | 0.01
 random_page_cost     | 4
 seq_page_cost        | 1
```

Cost = satır sayısı * cpu_tuple_cost + page sayısı * seq_page_cost

```
dellstore2=# SELECT relpages,reltuples FROM pg_class WHERE relname = 'customers';

 relpages | reltuples
----------+-----------
20000     | 488
```

Cost = 20000 * 0.01 + 488 * 1 = 688

```
dellstore2=# set seq_page_cost to 2;

dellstore2=# EXPLAIN ANALYZE SELECT * FROM customers;
                                                    QUERY PLAN
-------------------------------------------------------------------------------------------------------------------
 Seq Scan on customers  (cost=0.00..1176.00 rows=20000 width=1562) (actual time=0.015..5.532 rows=20000 loops=1)
 Total runtime: 7.985 ms
```

Cost = ?

- Başlangıç maliyeti ?

```
dellstore2=# explain analyze  SELECT max(customerid) FROM cust_hist;
                                                        QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------------------------------------
 Result  (cost=0.05..0.06 rows=1 width=0) (actual time=0.053..0.053 rows=1 loops=1)
   InitPlan 1 (returns $0)
     ->  Limit  (cost=0.00..0.05 rows=1 width=4) (actual time=0.044..0.044 rows=1 loops=1)
           ->  Index Scan Backward using ix_cust_hist_customerid on cust_hist  (cost=0.00..3027.10 rows=60048 width=4) (actual time=0.042..0.042 rows=1 loops=1)
                 Index Cond: (customerid IS NOT NULL)
 Total runtime: 0.128 ms


(6 rows)

dellstore2=# explain analyze  SELECT customerid FROM cust_hist;
                                      QUERY PLAN
---------------------------------------------------------------------------------------------------------
 Seq Scan on cust_hist  (cost=0.00..930.50 rows=60350 width=4) (actual time=0.024..18.548 rows=60350 loops=1)
 Total runtime: 24.854 ms
(2 rows)
```

IU: Bir sorgunun başlama maliyeti varsa tüm satırları çekmeden önce mutlaka bir işlem yapılıyordur. (sıralama gibi)

Sorgu örnekleri fsm.1

```
pgbench=# EXPLAIN ANALYZE SELECT * FROM pgbench_accounts ;
                                                  QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------
 Seq Scan on pgbench_accounts  (cost=0.00..641804.00 rows=15000000 width=97) (actual time=30.591..1511.705 rows=15000000 loops=1)
 Total runtime: 1963.580 ms

pgbench=# UPDATE pgbench_accounts SET bid=2;
UPDATE 15000000


pgbench=# EXPLAIN ANALYZE SELECT * FROM pgbench_accounts ;
                                                  QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------
 Seq Scan on pgbench_accounts  (cost=0.00..791804.00 rows=30000000 width=97) (actual time=126.774..19146.777 rows=15000000 loops=1)
 Total runtime: 19602.325 ms


pgbench=# SELECT * FROM pgstattuple('pgbench_accounts');
 table_len        | tuple_count | tuple_len      |   tuple_percent | dead_tuple_count | dead_tuple_len | dead_tuple_percent | free_space | free_percent
------------------+-------------+----------------+-----------------+------------------+----------------+--------------------+------------+--------------
 4028858368 | 15000000      |1815000000 |             45.05 |          15000000 |      1815000000 |                45.05 |   55087816 |        1.37


pgbench=# VACUUM pgbench_accounts ;
VACUUM

pgbench=# EXPLAIN ANALYZE SELECT * FROM pgbench_accounts ;
                                                  QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------
 Seq Scan on pgbench_accounts  (cost=0.00..641804.00 rows=15000000 width=97) (actual time=30.591..1511.705 rows=15000000 loops=1)
 Total runtime: 1963.580 ms

pgbench=# SELECT * FROM pgstattuple('pgbench_accounts');
 table_len        | tuple_count | tuple_len      | tuple_percent | dead_tuple_count | dead_tuple_len | dead_tuple_percent   | free_space     | free_percent
------------------+-------------+----------------+---------------+------------------+----------------+----------------------+----------------+--------------
 4028858368 |    15000000 | 1815000000 |             45.05 |                0 |              0 |                    0 | 1975087816 |        49.02
(1 row)
```

Sorgu örnekleri fsm.1

- Fazla FSM
- CLUSTER

•CLUSTER
  - Tüm tabloya LOCK koyacaktır.
  - INDEX gerekir
  - Disk
  - FSM 'yi temizler.

```
pgbench=# CLUSTER pgbench_accounts USING pgbench_accounts_pkey ;
CLUSTER

pgbench=# explain analyze select * from pgbench_accounts ;
                                                  QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------
 Seq Scan on pgbench_accounts  (cost=0.00..395902.00 rows=15000000 width=97) (actual time=0.033..1230.381 rows=15000000 loops=1)
 Total runtime: 1687.972 ms
(2 rows)


pgbench=# SELECT * FROM pgstattuple('pgbench_accounts');
 table_len      | tuple_count |     tuple_len  | tuple_percent | dead_tuple_count | dead_tuple_len | dead_tuple_percent | free_space | free_percent
----------------+-------------+----------------+---------------+------------------+----------------+--------------------+------------+--------------
 2014429184  |    15000000 |  1815000000 |          90.1 |                0 |              0 |                  0 |   27543928 |         1.37
(1 row)
```

Sorgu örnekleri scan1

- Sequential Scan
  - Hızlıdır.
  - Sırasızdır.
- Index Scan
  - Sequential
  - Daha çok I/O
  - Sıraldır
  - Tablonun **küçük** bir kısmı aranıyorsa kullanılır

```
pgbench=# EXPLAIN ANALYZE SELECT COUNT(aid) FROM pgbench_accounts;
                                                          QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------------------------
 Aggregate  (cost=433402.00..433402.01 rows=1 width=4) (actual time=2273.288..2273.288 rows=1 loops=1)
   ->  Seq Scan on pgbench_accounts  (cost=0.00..395902.00 rows=15000000 width=4) (actual time=0.014..1250.243 rows=15000000 loops=1)
 Total runtime: 2273.349 ms
(3 rows)

pgbench=# EXPLAIN ANALYZE SELECT MAX(aid) FROM pgbench_accounts;
                                                              QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 Result  (cost=0.04..0.05 rows=1 width=0) (actual time=0.051..0.051 rows=1 loops=1)
   InitPlan 1 (returns $0)
     ->  Limit  (cost=0.00..0.04 rows=1 width=4) (actual time=0.045..0.046 rows=1 loops=1)
           ->  Index Scan Backward using pgbench_accounts_pkey on pgbench_accounts  (cost=0.00..672930.90 rows=15000000 width=4) (actual
time=0.042..0.042 rows=1 loops=1)
                 Index Cond: (aid IS NOT NULL)
 Total runtime: 0.109 ms
```

Sorgu örnekleri scan2

- Bitmap Index Scan
  - Küçük parçaları farklı farklı yerlerde bulur.
  - Index kullanarak bitmap oluşturur. Sequential olarak tarar (atlayarak).
  - Seq. Scan den daha hızlıdır.
  - Koşul içeren sorgularda kullanılır. (aid = 2 OR aid = 3 vs..).

```
pgbench=# EXPLAIN ANALYZE SELECT * FROM pgbench_accounts WHERE aid = 3;
                                                        QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------------------------
 Index Scan using pgbench_accounts_pkey on pgbench_accounts  (cost=0.00..9.91 rows=1 width=97) (actual time=0.031..0.033 rows=1 loops=1)
   Index Cond: (aid = 3)
 Total runtime: 0.086 ms
(3 rows)

pgbench=# EXPLAIN ANALYZE SELECT * FROM pgbench_accounts WHERE aid = 3 OR aid =5;
                                                      QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------------------------
 Bitmap Heap Scan on pgbench_accounts  (cost=11.81..19.82 rows=2 width=97) (actual time=0.046..0.048 rows=2 loops=1)
   Recheck Cond: ((aid = 3) OR (aid = 5))
   ->  BitmapOr  (cost=11.81..11.81 rows=2 width=0) (actual time=0.037..0.037 rows=0 loops=1)
         ->  Bitmap Index Scan on pgbench_accounts_pkey  (cost=0.00..5.90 rows=1 width=0) (actual time=0.030..0.030 rows=1 loops=1)
               Index Cond: (aid = 3)
         ->  Bitmap Index Scan on pgbench_accounts_pkey  (cost=0.00..5.90 rows=1 width=0) (actual time=0.007..0.007 rows=1 loops=1)
               Index Cond: (aid = 5)
 Total runtime: 0.111 ms
(8 rows)
```

Sorgu örnekleri operators1

```
dellstore2=# EXPLAIN ANALYZE SELECT * FROM customers WHERE email LIKE
'ITHOMQJNYX@dell.com';
                                    QUERY PLAN
-----------------------------------------------------------------------------------------------------
 Seq Scan on customers  (cost=0.00..738.00 rows=1 width=268) (actual time=0.022..13.232 rows=1
loops=1)
   Filter: ((email)::text ~~ 'ITHOMQJNYX@dell.com'::text)
 Total runtime: 13.288 ms
(3 rows)


dellstore2=# CREATE INDEX customers_email_idx ON customers(email);
CREATE INDEX
dellstore2=# EXPLAIN ANALYZE SELECT * FROM customers WHERE email LIKE
'ITHOMQJNYX@dell.com';
                                              QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------
 Index Scan using customers_email_idx on customers  (cost=0.00..8.27 rows=1 width=268) (actual
time=0.099..0.101 rows=1 loops=1)
   Index Cond: ((email)::text = 'ITHOMQJNYX@dell.com'::text)
   Filter: ((email)::text ~~ 'ITHOMQJNYX@dell.com'::text)
 Total runtime: 0.161 ms
(4 rows)
```

Sorgu örnekleri operators2

```
dellstore2=# EXPLAIN ANALYZE SELECT * FROM customers WHERE email LIKE 'ITHOMQJNYX@%';
                                                  QUERY PLAN
---------------------------------------------------------------------------------------------------------
 Seq Scan on customers  (cost=0.00..738.00 rows=2 width=268) (actual time=0.025..13.207 rows=1
loops=1)
   Filter: ((email)::text ~~ 'ITHOMQJNYX@%'::text)
 Total runtime: 13.266 ms
(3 rows)

dellstore2=# CREATE INDEX customers_email_ops_idx ON customers(email varchar_pattern_ops);
CREATE INDEX

dellstore2=# EXPLAIN ANALYZE SELECT * FROM customers WHERE email LIKE 'ITHOMQJNYX@%';
                                                        QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------
 Index Scan using customers_email_ops_idx on customers  (cost=0.00..8.27 rows=2 width=268) (actual
time=0.081..0.083 rows=1 loops=1)
   Index Cond: (((email)::text ~>=~ 'ITHOMQJNYX@'::text) AND ((email)::text ~<~ 'ITHOMQJNYXA'::text))
   Filter: ((email)::text ~~ 'ITHOMQJNYX@%'::text)
 Total runtime: 0.149 ms
(4 rows)
```

Tamamdır yeniden yazalım.

Uygulamalı olarak büyük bir sorgunun yeniden yazımı.

Notlar

- SQL cümlecikleri
- Analiz araçları (pgFouine ? pg_stat_statements ?)
- Haberdar olun.
- AUTOVACUUM & VACUUM
- AUTOANALYZE & ANALYZE
- Tablo boyuları
- Test veritabanı
- Kullanılmayandan kurtulun
- **AUTO_COMMIT KULLANMAYIN!** - psql kullanıcıları için
- Psql kullanın.
- Materialize VIEW ?
- irc.freenode.net #postgresql sorun

Teşekkürler

SORULAR ?