DATABASE PERFORMANCE TUNING

11/7/2013

Tugas I IF3140—Manajemen Basis Data

oleh

Arief Rahman	13511020
Muhammad Nassirudin	13511044
Rifki Afina Putri	13511066

DATABASE PERFORMANCE TUNING

TRANSACTION TUNING DAN DATABASE DESIGN TUNING

TRANSACTION TUNING

Analisis dilakukan menggunakan fitur `explain verbose` dari ProgreSQL terhadap 3 query yang menghasilkan keluaran yang sama (lihat gambar 1). Berdasarkan hasil yang didapat, penggunaan query 1 dan 2 menunjukkan hasil yang sama, sementara query 3 menunjukkan hasil yang berbeda. Hasil dari ketiga query diberikan pada gambar 2 dan 3.

```
select nama_benda_seni
from benda_seni
natural join lukisan
natural join seniman
natural join dipamerkan
natural join pagelaran
where seniman.negara_asal = 'Italy'
and pagelaran.bulan = 'April'
```

```
select nama_benda_seni
from benda_seni
natural join lukisan
natural join (select * from seniman
where negara_asal = 'Italy') as
seniman_italy
natural join dipamerkan
natural join (select * from
pagelaran where bulan = 'April') as
pagelaran_april
```

```
select nama_benda_seni
from lukisan
natural join
(select * from seniman
natural join dipamerkan
natural join pagelaran
natural join benda_seni
where negara_asal like 'Italy' and
bulan like 'April') as
benda_italy_april
```

Gambar 1. (a) query 1 (kiri atas), (b) query 2 (kanan atas), dan (c) query 3 (bawah)

```
Nested Loop  (cost=28.89..50.31 rows=1 width=12)
  Output: benda_seni.nama_benda_seni
  ->  Hash Join  (cost=28.75..49.54 rows=4 width=23)
        Output: benda_seni.nama_benda_seni, dipamerkan.nama_pagelaran
        Hash Cond: ((dipamerkan.nama_benda_seni)::text = (benda_seni.nama_benda_seni)::text)
        ->  Seq Scan on public.dipamerkan  (cost=0.00..17.00 rows=1000 width=23)
              Output: dipamerkan.nama_pagelaran, dipamerkan.nama_benda_seni
        ->  Hash  (cost=28.72..28.72 rows=2 width=23)
              Output: benda_seni.nama_benda_seni, lukisan.nama_benda_seni
              ->  Hash Join  (cost=23.27..28.72 rows=2 width=23)
                    Output: benda_seni.nama_benda_seni, lukisan.nama_benda_seni
                    Hash Cond: ((lukisan.nama_benda_seni)::text = (benda_seni.nama_benda_seni)::text)
                    ->  Seq Scan on public.lukisan  (cost=0.00..4.50 rows=250 width=11)
                          Output: lukisan.nama_benda_seni, lukisan.kategori
                    ->  Hash  (cost=23.22..23.22 rows=4 width=12)
                          Output: benda_seni.nama_benda_seni
                          ->  Hash Join  (cost=10.30..23.22 rows=4 width=12)
                                Output: benda_seni.nama_benda_seni
                                Hash Cond: ((benda_seni.nama_seniman)::text = (seniman.nama_seniman)::text)
                                ->  Seq Scan on public.benda_seni  (cost=0.00..11.00 rows=500 width=18)
                                      Output: benda_seni.nama_benda_seni, benda_seni.tahun_pembuatan,
benda_seni.deskripsi, benda_seni.nama_seniman
                                ->  Hash  (cost=10.25..10.25 rows=4 width=6)
                                      Output: seniman.nama_seniman
                                      ->  Seq Scan on public.seniman  (cost=0.00..10.25 rows=4 width=6)
                                            Output: seniman.nama_seniman
                                            Filter: ((seniman.negara_asal)::text = 'Italy'::text)
  ->  Index Scan using pagelaran_pkey on public.pagelaran  (cost=0.14..0.18 rows=1 width=11)
        Output: pagelaran.nama_pagelaran, pagelaran.bulan
        Index Cond: ((pagelaran.nama_pagelaran)::text = (dipamerkan.nama_pagelaran)::text)
        Filter: ((pagelaran.bulan)::text = 'April'::text)
```

Gambar 2. Hasil Pemanggilan Query 1 dan 2

```
Nested Loop  (cost=28.89..50.70 rows=1 width=11)
  Output: lukisan.nama_benda_seni
  ->  Hash Join  (cost=28.75..49.54 rows=6 width=22)
        Output: lukisan.nama_benda_seni, dipamerkan.nama_pagelaran
        Hash Cond: ((dipamerkan.nama_benda_seni)::text = (benda_seni.nama_benda_seni)::text)
        ->  Seq Scan on public.dipamerkan  (cost=0.00..17.00 rows=1000 width=23)
              Output: dipamerkan.nama_pagelaran, dipamerkan.nama_benda_seni
        ->  Hash  (cost=28.72..28.72 rows=2 width=23)
              Output: lukisan.nama_benda_seni, benda_seni.nama_benda_seni
              ->  Hash Join  (cost=23.27..28.72 rows=2 width=23)
                    Output: lukisan.nama_benda_seni, benda_seni.nama_benda_seni
                    Hash Cond: ((lukisan.nama_benda_seni)::text = (benda_seni.nama_benda_seni)::text)
                    ->  Seq Scan on public.lukisan  (cost=0.00..4.50 rows=250 width=11)
                          Output: lukisan.nama_benda_seni, lukisan.kategori
                    ->  Hash  (cost=23.22..23.22 rows=4 width=12)
                          Output: benda_seni.nama_benda_seni
                          ->  Hash Join  (cost=10.30..23.22 rows=4 width=12)
                                Output: benda_seni.nama_benda_seni
                                Hash Cond: ((benda_seni.nama_seniman)::text = (seniman.nama_seniman)::text)
                                ->  Seq Scan on public.benda_seni  (cost=0.00..11.00 rows=500 width=18)
                                      Output: benda_seni.nama_benda_seni, benda_seni.tahun_pembuatan,
benda_seni.deskripsi, benda_seni.nama_seniman
                                ->  Hash  (cost=10.25..10.25 rows=4 width=6)
                                      Output: seniman.nama_seniman
                                      ->  Seq Scan on public.seniman  (cost=0.00..10.25 rows=4 width=6)
                                            Output: seniman.nama_seniman
                                            Filter: ((seniman.negara_asal)::text ~~ 'Italy'::text)
  ->  Index Scan using pagelaran_pkey on public.pagelaran  (cost=0.14..0.18 rows=1 width=11)
        Output: pagelaran.nama_pagelaran, pagelaran.bulan
        Index Cond: ((pagelaran.nama_pagelaran)::text = (dipamerkan.nama_pagelaran)::text)
        Filter: ((pagelaran.bulan)::text ~~ 'April'::text)
```

Gambar 3. Hasil Pemanggilan Query 3

Dari hasil analisis di atas, dapat terlihat bahwa jumlah cost pada query 1, query 2, dan query 3 tidak berbeda terlalu jauh. Namun, cost pada query 3 lebih besar dibandingkan dengan query lainnya. Karena setelah optimasi jumlah baris yang dihasilkan setelah join lebih besar dibandingkan dengan query 1 dan 2.

Sedangkan query 1 dan 2 memiliki hasil yang serupa karena query yang ditulis ulang oleh DBMS sama.

DATABASE DESIGN TUNING

Berdasarkan analisis dari query berikut (gambar 4), dipilih beberapa tuning terhadap skema database, yaitu (1) horizontal splitting, (2) collapsed table, dan (3) adding derived and redundant column. Masing-masing cara yang dipilih dijelaskan berikut ini.

```
select distinct nama_benda_seni,
negara_asal
from benda_seni
natural join seniman
natural join dipamerkan
natural join pagelaran
where negara_asal in
(select negara_asal from seniman
group by negara_asal
having count(negara_asal) > 4)
and tahun pembuatan > 2005;
```

Gambar 4. Query yang dipakai

Berikut ini adalah hasil analisis yang dilakukan menggunakan fitur `explain verbose` dari ProgreSQL sebelum tuning pada skema:

```
HashAggregate  (cost=151.81..160.81 rows=900 width=23) (actual time=4.360..4.373 rows=23 loops=1)
  ->  Hash Join  (cost=63.68..147.31 rows=900 width=23) (actual time=1.904..4.140 rows=198 loops=1)
        Hash Cond: ((dipamerkan.nama_pagelaran)::text = (pagelaran.nama_pagelaran)::text)
        ->  Hash Join  (cost=56.06..127.31 rows=900 width=34) (actual time=1.601..3.642 rows=198
loops=1)
              Hash Cond: ((dipamerkan.nama_benda_seni)::text = (benda_seni.nama_benda_seni)::text)
              ->  Seq Scan on dipamerkan  (cost=0.00..51.00 rows=3000 width=23) (actual time=0.016..0.806
rows=3000 loops=1)
              ->  Hash  (cost=54.18..54.18 rows=150 width=23) (actual time=1.568..1.568 rows=28 loops=1)
                    Buckets: 1024  Batches: 1  Memory Usage: 2kB
                    ->  Hash Join  (cost=39.31..54.18 rows=150 width=23) (actual time=1.157..1.538 rows=28
loops=1)
                          Hash Cond: ((benda_seni.nama_seniman)::text = (seniman.nama_seniman)::text)
                          ->  Seq Scan on benda_seni  (cost=0.00..12.25 rows=300 width=18) (actual
time=0.013..0.263 rows=300 loops=1)
                                Filter: (tahun_pembuatan > 2005)
                                Rows Removed by Filter: 200
                          ->  Hash  (cost=36.18..36.18 rows=250 width=17) (actual time=1.131..1.131 rows=54
loops=1)
                                Buckets: 1024  Batches: 1  Memory Usage: 3kB
                                ->  Hash Semi Join  (cost=19.02..36.18 rows=250 width=17) (actual
time=0.759..1.086 rows=54 loops=1)
                                      Hash Cond: ((seniman.negara_asal)::text = (seniman_1.negara_asal)::text)
                                      ->  Seq Scan on seniman  (cost=0.00..9.00 rows=500 width=17) (actual
time=0.008..0.110 rows=500 loops=1)
                                      ->  Hash  (cost=16.34..16.34 rows=215 width=11) (actual time=0.737..0.737
rows=10 loops=1)
                                            Buckets: 1024  Batches: 1  Memory Usage: 1kB
                                            ->  HashAggregate  (cost=11.50..14.19 rows=215 width=11) (actual
time=0.664..0.727 rows=10 loops=1)
                                                  Filter: (count(seniman_1.negara_asal) > 4)
                                                  Rows Removed by Filter: 205
                                                  ->  Seq Scan on seniman seniman_1  (cost=0.00..9.00 rows=500
width=11) (actual time=0.007..0.112 rows=500 loops=1)
        ->  Hash  (cost=4.50..4.50 rows=250 width=11) (actual time=0.282..0.282 rows=250 loops=1)
              Buckets: 1024  Batches: 1  Memory Usage: 9kB
              ->  Seq Scan on pagelaran  (cost=0.00..4.50 rows=250 width=11) (actual time=0.013..0.109
rows=250 loops=1)
Total runtime: 4.703 ms
```

Gambar 5. Hasil analisis query sebelum tuning pada skema

Cara Pertama

Proses

Horizontal splitting. Tabel `benda_seni` dipisah menjadi 2 dengan memishkan yang memiliki atribut `tahun_pembuatan` yang lebih dari tahun 2005 dan yang kurang dari atau setahun dengan 2005. Pemisahan dilakukan dengan memanfaatkan prinsip inheritance, yaitu dengan membuat tabel anak dari

tabel benda_seni yang sudah ada. Oleh karena itu, tidak ada perubahan pada query yang bersangkutan.

Berikut ini merupakan syntax yang dieksekusi untuk tuning menggunakan horizontal splitting:

```
-- Horizontal Splitting
-- Create the partitions
CREATE TABLE benda_seni_more_y2005 (
      PRIMARY KEY (nama_benda_seni),
      FOREIGN KEY (nama_seniman) REFERENCES seniman (nama_seniman),
      CHECK (tahun_pembuatan > 2005)
) INHERITS (benda_seni);

CREATE TABLE benda_seni_y2005_less (
      PRIMARY KEY (nama_benda_seni),
      FOREIGN KEY (nama_seniman) REFERENCES seniman (nama_seniman),
      CHECK (tahun_pembuatan <= 2005)
) INHERITS (benda_seni);

-- Inserting data
INSERT INTO benda_seni_more_y2005 SELECT * FROM benda_seni WHERE
tahun_pembuatan > 2005;

INSERT INTO benda_seni_y2005_less SELECT * FROM benda_seni WHERE
tahun_pembuatan <= 2005;

-- Deleting tables
DROP TABLE benda_seni_more_y2005 CASCADE;
DROP TABLE benda_seni_y2005_less CASCADE;
```

Gambar 6. Syntax untuk horizontal splitting

Alasan

Pemisahan tersebut membuat pencarian tahun_pembuatan yang sesuai dilakukan lebih efisien.

Hasil

```
HashAggregate  (cost=250.60..276.92 rows=2632 width=23) (actual time=2.563..2.569 rows=23
loops=1)
  ->  Hash Join  (cost=135.43..237.44 rows=2632 width=23) (actual time=1.913..2.429 rows=396
loops=1)
        Hash Cond: ((dipamerkan.nama_pagelaran)::text = (pagelaran.nama_pagelaran)::text)
        ->  Hash Join  (cost=127.81..193.63 rows=2632 width=34) (actual time=1.787..2.192 rows=396
loops=1)
              Hash Cond: ((benda_seni.nama_benda_seni)::text = (dipamerkan.nama_benda_seni)::text)
              ->  Hash Join  (cost=39.31..64.56 rows=300 width=23) (actual time=0.448..0.758 rows=56
loops=1)
                    Hash Cond: ((benda_seni.nama_seniman)::text = (seniman.nama_seniman)::text)
                    ->  Append  (cost=0.00..20.00 rows=600 width=18) (actual time=0.007..0.229 rows=600
loops=1)
                          ->  Seq Scan on benda_seni  (cost=0.00..12.25 rows=300 width=18) (actual
time=0.007..0.121 rows=300 loops=1)
                                Filter: (tahun_pembuatan > 2005)
                                Rows Removed by Filter: 200
                          ->  Seq Scan on benda_seni_more_y2005  (cost=0.00..7.75 rows=300 width=18) (actual
time=0.004..0.074 rows=300 loops=1)
                                Filter: (tahun_pembuatan > 2005)
                    ->  Hash  (cost=36.18..36.18 rows=250 width=17) (actual time=0.435..0.435 rows=54
loops=1)
                          Buckets: 1024  Batches: 1  Memory Usage: 3kB
                          ->  Hash Semi Join  (cost=19.02..36.18 rows=250 width=17) (actual time=0.299..0.418
rows=54 loops=1)
                                Hash Cond: ((seniman.negara_asal)::text = (seniman_1.negara_asal)::text)
                                ->  Seq Scan on seniman  (cost=0.00..9.00 rows=500 width=17) (actual
time=0.006..0.041 rows=500 loops=1)
                                ->  Hash  (cost=16.34..16.34 rows=215 width=11) (actual time=0.286..0.286
rows=10 loops=1)
                                      Buckets: 1024  Batches: 1  Memory Usage: 1kB
                                      ->  HashAggregate  (cost=11.50..14.19 rows=215 width=11) (actual
time=0.258..0.282 rows=10 loops=1)
                                            Filter: (count(seniman_1.negara_asal) > 4)
                                            Rows Removed by Filter: 205
                                            ->  Seq Scan on seniman seniman_1  (cost=0.00..9.00 rows=500 width=11)
(actual time=0.003..0.050 rows=500 loops=1)
              ->  Hash  (cost=51.00..51.00 rows=3000 width=23) (actual time=1.332..1.332 rows=3000
loops=1)
                    Buckets: 1024  Batches: 1  Memory Usage: 141kB
                    ->  Seq Scan on dipamerkan  (cost=0.00..51.00 rows=3000 width=23) (actual
time=0.008..0.487 rows=3000 loops=1)
        ->  Hash  (cost=4.50..4.50 rows=250 width=11) (actual time=0.119..0.119 rows=250 loops=1)
              Buckets: 1024  Batches: 1  Memory Usage: 9kB
              ->  Seq Scan on pagelaran  (cost=0.00..4.50 rows=250 width=11) (actual time=0.012..0.046
rows=250 loops=1)
Total runtime: 2.748 ms
```

Gambar 7. Hasil horizontal splitting

Dari hasil di atas, dapat terlihat bahwa berdasarkan prediksi query explain, cost pada tabel yang telah di-*split* lebih besar, namun pada kenyataannya, hasil eksekusi lebih cepat dibandingkan dengan hasil eksekusi pada skema awal.

Cara Kedua

Proses

Collapsed table. Dilakukan penggabungan antara `tabel benda_seni`, `pagelaran`, `dipamerkan`, dan `seniman` menjadi sebuah tabel baru bernama `full_info_benda_seni`. Namun, perlu adanya perubahan query terkait dengan penggabungan tabel ini. Berikut ini adalah perubahan query yang dilakukan:

```
select distinct nama_benda_seni,
negara_asal
from full_info_benda_seni
where negara_asal in
(select negara_asal from seniman
group by negara_asal
having count(negara_asal) > 4)
     and tahun_pembuatan > 2005;
```

Gambar 8. Perubahan query untuk collapsed table

Berikut ini merupakan syntax yang digunakan untuk tuning menggunakan collapsed table:

```
-- Table Collapsing
-- Creating collapsed table
CREATE TABLE full_info_benda_seni (
      nama_benda_seni VARCHAR(50),
      tahun_pembuatan INTEGER,
      deskripsi VARCHAR(200),
      nama_seniman VARCHAR(50),
      nama_pagelaran VARCHAR(50),
      bulan VARCHAR(10),
      negara_asal VARCHAR(50)
);

-- Inserting data to table
INSERT INTO full_info_benda_seni SELECT nama_benda_seni, tahun_pembuatan,
deskripsi, nama_seniman, nama_pagelaran, bulan, negara_asal FROM benda_seni
NATURAL JOIN seniman NATURAL JOIN dipamerkan NATURAL JOIN pagelaran;

-- Deleting table
DROP TABLE full_info_benda_seni;
```

Gambar 9. Syntax untuk collapsed table

Alasan

Penggabungan tabel dilakukan untuk mengurangi natural join antara tabel `seniman`, `dipamerkan`, dan `pagelaran` yang terdapat pada query sebelumnya. Sehingga, tidak perlu lagi dilakukan join saat query yang baru dieksekusi.

Hasil

```
HashAggregate  (cost=57.73..59.77 rows=204 width=22) (actual time=2.153..2.165 rows=23 loops=1)
  ->  Hash Semi Join  (cost=19.02..56.24 rows=298 width=22) (actual time=1.124..2.060 rows=66
loops=1)
        Hash Cond: ((full_info_benda_seni.negara_asal)::text = (seniman.negara_asal)::text)
        ->  Seq Scan on full_info_benda_seni  (cost=0.00..27.50 rows=595 width=22) (actual
time=0.030..0.578 rows=595 loops=1)
              Filter: (tahun_pembuatan > 2005)
              Rows Removed by Filter: 405
        ->  Hash  (cost=16.34..16.34 rows=215 width=11) (actual time=1.061..1.061 rows=10 loops=1)
              Buckets: 1024  Batches: 1  Memory Usage: 1kB
              ->  HashAggregate  (cost=11.50..14.19 rows=215 width=11) (actual time=0.968..1.049 rows=10
loops=1)
                    Filter: (count(seniman.negara_asal) > 4)
                    Rows Removed by Filter: 205
                    ->  Seq Scan on seniman  (cost=0.00..9.00 rows=500 width=11) (actual time=0.023..0.168
rows=500 loops=1)
Total runtime: 2.362 ms
```

Gambar 10. Hasil collapsed table

Dari hasil analisis di atas, dapat terlihat bahwa tidak perlu dilakukan lagi penggabungan antara tabel `seniman`, `dipamerkan`, dan `pagelaran`. Hal ini berpengaruh pada cost yang lebih sedikit bila dibandingkan dengan hasil query sebelum dilakukan tuning. Sehingga cost dan waktu eksekusinya pun lebih sedikit.

Cara Ketiga

Proses

Adding derived and redundant column. Pada cara ini dilakukan pembuatan tabel baru bernama count_negara_asal yang berisi negara asal dan jumlah negara asal. Perlu adanya perubahan query terkait dengan prosesn tuning ini. Berikut ini perubahan query yang dilakukan:

```
select distinct nama_benda_seni,
negara_asal
from benda_seni
natural join seniman
natural join dipamerkan
natural join pagelaran
where negara_asal in
(select negara_asal from
count_negara_asal
where jumlah_negara_asal > 4)
     and tahun_pembuatan > 2005;
```

Gambar 11. Perubahan query untuk adding derived and redundant column

Berikut ini syntax yang dieksekusi untuk melakukan proses ini:

```
-- Derived and Redundant Columns
-- Creating table
CREATE TABLE count_negara_asal (
      negara_asal VARCHAR(50) PRIMARY KEY,
      jumlah_negara_asal INTEGER
);

-- Inserting data
INSERT INTO count_negara_asal SELECT negara_asal, count(negara_asal) AS
jumlah_negara_asal
FROM seniman GROUP BY negara_asal;

-- Deleting table
DROP TABLE count_negara_asal;
```

Gambar 12. Syntax untuk adding derived and redundant column

Alasan

Proses ini dilakukan untuk menghilangkan fungsi agregat count yang memiliki cost tinggi.

Hasil

```
HashAggregate  (cost=122.98..125.43 rows=245 width=23) (actual time=5.984..5.995 rows=23 loops=1)
  ->  Hash Join  (cost=46.17..121.75 rows=245 width=23) (actual time=2.209..5.493 rows=396 loops=1)
        Hash Cond: ((dipamerkan.nama_pagelaran)::text = (pagelaran.nama_pagelaran)::text)
        ->  Hash Join  (cost=38.55..110.76 rows=245 width=34) (actual time=1.817..4.663 rows=396
loops=1)
              Hash Cond: ((dipamerkan.nama_benda_seni)::text = (benda_seni.nama_benda_seni)::text)
              ->  Seq Scan on dipamerkan  (cost=0.00..51.00 rows=3000 width=23) (actual time=0.020..0.964
rows=3000 loops=1)
              ->  Hash  (cost=38.20..38.20 rows=28 width=23) (actual time=1.777..1.777 rows=56 loops=1)
                    Buckets: 1024  Batches: 1  Memory Usage: 3kB
                    ->  Hash Join  (cost=15.67..38.20 rows=28 width=23) (actual time=0.675..1.696 rows=56
loops=1)
                          Hash Cond: ((benda_seni.nama_seniman)::text = (seniman.nama_seniman)::text)
                          ->  Append  (cost=0.00..20.00 rows=600 width=18) (actual time=0.020..0.759
rows=600 loops=1)
                                ->  Seq Scan on benda_seni  (cost=0.00..12.25 rows=300 width=18) (actual
time=0.018..0.369 rows=300 loops=1)
                                      Filter: (tahun_pembuatan > 2005)
                                      Rows Removed by Filter: 200
                                ->  Seq Scan on benda_seni_more_y2005  (cost=0.00..7.75 rows=300 width=18)
(actual time=0.017..0.270 rows=300 loops=1)
                                      Filter: (tahun_pembuatan > 2005)
                          ->  Hash  (cost=15.38..15.38 rows=23 width=17) (actual time=0.639..0.639 rows=54
loops=1)
                                Buckets: 1024  Batches: 1  Memory Usage: 3kB
                                ->  Hash Semi Join  (cost=4.81..15.38 rows=23 width=17) (actual time=0.153..0.585
rows=54 loops=1)
                                      Hash Cond: ((seniman.negara_asal)::text = (count_negara_asal.negara_asal)::text)
                                      ->  Seq Scan on seniman  (cost=0.00..9.00 rows=500 width=17) (actual
time=0.017..0.149 rows=500 loops=1)
                                      ->  Hash  (cost=4.69..4.69 rows=10 width=11) (actual time=0.120..0.120
rows=10 loops=1)
                                            Buckets: 1024  Batches: 1  Memory Usage: 1kB
                                            ->  Seq Scan on count_negara_asal  (cost=0.00..4.69 rows=10 width=11)
(actual time=0.033..0.108 rows=10 loops=1)
                                                  Filter: (jumlah_negara_asal > 4)
                                                  Rows Removed by Filter: 205
        ->  Hash  (cost=4.50..4.50 rows=250 width=11) (actual time=0.358..0.358 rows=250 loops=1)
              Buckets: 1024  Batches: 1  Memory Usage: 9kB
              ->  Seq Scan on pagelaran  (cost=0.00..4.50 rows=250 width=11) (actual time=0.020..0.141
rows=250 loops=1)
Total runtime: 6.360 ms
```

Gambar 13. Hasil adding derived and redundant column

Dari hasil query analysis di atas, dapat terlihat bahwa cost setelah dilakukan perubahan skema menjadi lebih sedikit dibandingkan dengan sebelum dilakukan perubahan skema.