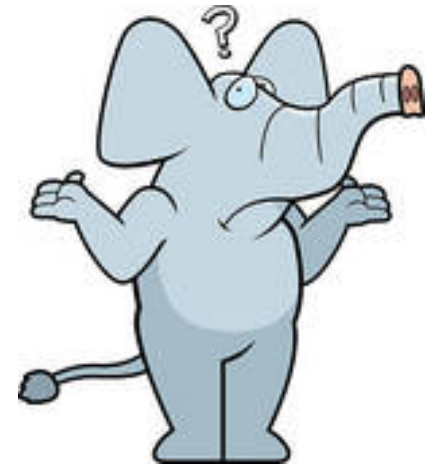


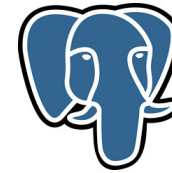
Conception Avancée de Bases de Données

PostgreSQL
PGAdminIII
Explain

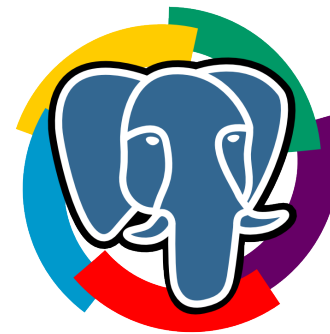


Traduction en cours

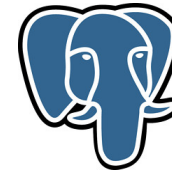
Numbers



- The first set of numbers (**cost=0.00..14.80**)
 - is an estimate of how "expensive" this operation will be.
- « Expensive" is measured in terms of disk reads



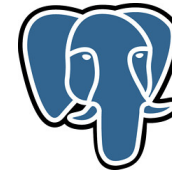
Jointure entre R et S



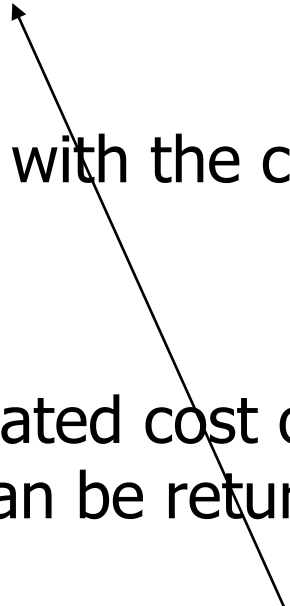

```
postgres=# explain select * from r,s;  
          QUERY PLAN
```

```
-----  
Nested Loop (cost=15.28..4638.08 rows=230400 width=276)  
-> Seq Scan on r (cost=0.00..14.80 rows=480 width=138)  
-> Materialize (cost=15.28..20.08 rows=480 width=138)  
    -> Seq Scan on s (cost=0.00..14.80 rows=480 width=138)  
(4 lignes)
```

Numbers



Seq Scan on r (cost=0.00..14.80 rows=480 width=138)

- Two numbers are associated with the cost, separated by two periods.

- The first number is the estimated cost of startup (the time spent before the first tuple can be returned) .
- The second number is the estimated *total* cost that the query will incur to completely execute.

The time spent before the first tuple can be returned

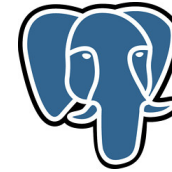


```
emmanuelfuchs — psql — 64x6
cavbd=> EXPLAIN SELECT COUNT(*) FROM country;
Aggregate  (cost=7.99..8.00 rows=1 width=0)
  -> Seq Scan on country  (cost=0.00..7.39 rows=239 width=0)
cavbd=> █
```

It's important to note that the cost of an upper-level node includes the cost of all its child nodes.

<http://www.postgresql.org/docs/8.1/static/performance-tips.html#USING-EXPLAIN>

Jointure entre R et S



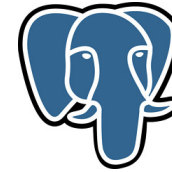
```
postgres=# explain select * from r,s;  
          QUERY PLAN
```

```
-----  
Nested Loop (cost=15.28..4638.08 rows=230400 width=276)  
-> Seq Scan on r (cost=0.00..14.80 rows=480 width=138)  
-> Materialize (cost=15.28..20.08 rows=480 width=138)  
    -> Seq Scan on s (cost=0.00..14.80 rows=480 width=138)  
(4 lignes)
```

A materialize node means the output of the node below it in the tree is materialized into memory before the upper node is executed. The materialized output will be rescanned in memory. The intermediate result is stored in memory.

A materialized view is a table that actually contains rows, but behaves like a view. That is, the data in the table changes when the data in the underlying tables changes.

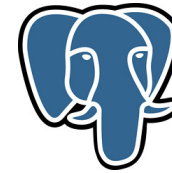
Numbers



Seq Scan on r (cost=0.00..14.80 rows=480 width=138)

- The second data item in the cost estimate (rows=480) shows how many rows PostgreSQL expects to return from this operation.
- The final data item (width=138) is an estimate of the width, in bytes, of the average row in the result set.

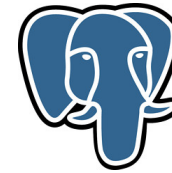
Jointure entre R et S



```
postgres=# explain select * from r,s;  
          QUERY PLAN
```

```
-----  
Nested Loop (cost=15.28..4638.08 rows=230400 width=276)  
-> Seq Scan on r (cost=0.00..14.80 rows=480 width=138)  
-> Materialize (cost=15.28..20.08 rows=480 width=138)  
    -> Seq Scan on s (cost=0.00..14.80 rows=480 width=138)  
(4 lignes)
```


Jointure entre R et S : Analyse



```
postgres=# explain analyse select * from r,s;  
          QUERY PLAN
```

```
-----  
Nested Loop (cost=15.28..4638.08 rows=230400 width=276) (actual time=0.056..0.956 rows=100 loops=1)  
-> Seq Scan on r (cost=0.00..14.80 rows=480 width=138) (actual time=0.011..0.036 rows=10 loops=1)  
-> Materialize (cost=15.28..20.08 rows=480 width=138) (actual time=0.005..0.035 rows=10 loops=10)  
    -> Seq Scan on s (cost=0.00..14.80 rows=480 width=138) (actual time=0.005..0.030 rows=10 loops=1)  
Total runtime: 1.297 ms  
(5 lignes)
```

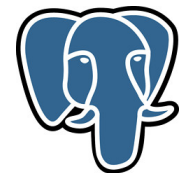
the "loops" value reports the total number of executions of the node, and the actual time and rows values shown are averages per-execution

Jointure entre R et S : Analyse et verbose

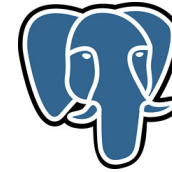


```
postgres=# explain analyse verbose select * from r,s;  
QUERY PLAN
```

```
-----  
Nested Loop (cost=15.28..4638.08 rows=230400 width=276) (actual time=0.055..0.941  
rows=100 loops=1)  
  Output: r."char", s."char"  
    -> Seq Scan on r (cost=0.00..14.80 rows=480 width=138) (actual time=0.012..0.036 rows=10  
loops=1)  
      Output: r."char"  
        -> Materialize (cost=15.28..20.08 rows=480 width=138) (actual time=0.005..0.033 rows=10  
loops=10)  
          Output: s."char"  
            -> Seq Scan on s (cost=0.00..14.80 rows=480 width=138) (actual time=0.005..0.031  
rows=10 loops=1)  
              Output: s."char"  
Total runtime: 1.263 ms  
(9 lignes)
```



Select with ORDER BY



```
postgres=# explain analyse verbose select * from r ORDER BY char;  
QUERY PLAN
```

```
-----  
Sort (cost=36.18..37.38 rows=480 width=138) (actual time=0.166..0.190 rows=10 loops=1)
```

```
Output: "char"
```

```
Sort Key: "char"
```

```
Sort Method: quicksort Memory: 17kB
```

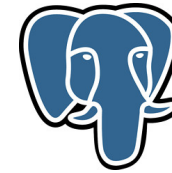
```
-> Seq Scan on r (cost=0.00..14.80 rows=480 width=138) (actual time=0.007..0.032 rows=10 loops=1)
```

```
Output: "char"
```

```
Total runtime: 0.272 ms
```

```
(7 lignes)
```

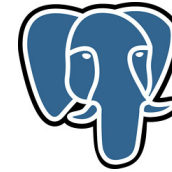
Jointure entre R et S avec ORDER BY



```
postgres=# explain analyse verbose select * from r,s order by R.char;  
QUERY PLAN
```

```
-----  
Sort (cost=85009.56..85585.56 rows=230400 width=276) (actual time=1.508..1.740 rows=100 loops=1)  
  Output: r."char", s."char"  
  Sort Key: r."char"  
  Sort Method: quicksort Memory: 20kB  
-> Nested Loop (cost=15.28..4638.08 rows=230400 width=276) (actual time=0.052..0.947 rows=100 loops=1)  
  Output: r."char", s."char"  
    -> Seq Scan on r (cost=0.00..14.80 rows=480 width=138) (actual time=0.012..0.037 rows=10 loops=1)  
      Output: r."char"  
    -> Materialize (cost=15.28..20.08 rows=480 width=138) (actual time=0.005..0.034 rows=10 loops=10)  
      Output: s."char"  
        -> Seq Scan on s (cost=0.00..14.80 rows=480 width=138) (actual time=0.005..0.030 rows=10 loops=1)  
          Output: s."char"  
Total runtime: 2.050 ms  
(13 lignes)
```

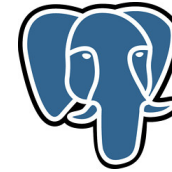
Select *



```
postgres=# explain analyse verbose select * from r;  
          QUERY PLAN
```

```
-----  
Seq Scan on r (cost=0.00..14.80 rows=480 width=138) (actual time=0.012..0.037 rows=10 loops=1)  
  Output: "char"  
Total runtime: 0.135 ms  
(3 lignes)
```

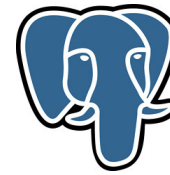
Selection sur une constante



```
postgres=# explain analyse verbose select * from r where char='B';  
          QUERY PLAN
```

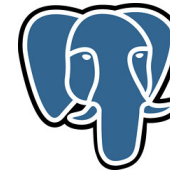
```
-----  
Seq Scan on r (cost=0.00..16.00 rows=2 width=138) (actual time=0.016..0.025 rows=2 loops=1)  
  Output: "char"  
  Filter: (("char")::text = 'B'::text)  
Total runtime: 0.080 ms  
(4 lignes)
```

SELECT DISTINCT



Select distinct S.char
From R,S
Where s.char=r.char;

Explain



```
postgres=# select distinct S.char from R,S where s.char=r.char;
```

```
char
```

```
-----
```

```
B
```

```
E
```

```
K
```

```
U
```

```
V
```

```
(5 lignes)
```

```
postgres=# explain select distinct S.char from R,S where s.char=r.char;
```

```
QUERY PLAN
```

```
-----
```

```
Unique (cost=72.35..94.91 rows=200 width=138)
```

```
-> Merge Join (cost=72.35..92.03 rows=1152 width=138)
```

```
    Merge Cond: ((r."char")::text = (s."char")::text)
```

```
    -> Sort (cost=36.18..37.38 rows=480 width=138)
```

```
        Sort Key: r."char"
```

```
        -> Seq Scan on r (cost=0.00..14.80 rows=480 width=138)
```

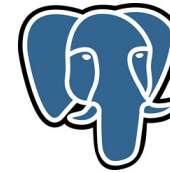
```
    -> Sort (cost=36.18..37.38 rows=480 width=138)
```

```
        Sort Key: s."char"
```

```
        -> Seq Scan on s (cost=0.00..14.80 rows=480 width=138)
```

```
(9 lignes)
```


Explain Analyse



```
postgres=# explain select distinct S.char from R,S where s.char=r.char;  
          QUERY PLAN
```

```
-----  
Unique  (cost=72.35..94.91 rows=200 width=138)  
-> Merge Join (cost=72.35..92.03 rows=1152 width=138)  
    Merge Cond: ((r."char")::text = (s."char")::text)  
    -> Sort (cost=36.18..37.38 rows=480 width=138)  
        Sort Key: r."char"  
        -> Seq Scan on r (cost=0.00..14.80 rows=480 width=138)  
    -> Sort (cost=36.18..37.38 rows=480 width=138)  
        Sort Key: s."char"  
        -> Seq Scan on s (cost=0.00..14.80 rows=480 width=138)  
(9 lignes)
```

```
postgres=# explain analyse select distinct S.char from R,S where s.char=r.char;  
          QUERY PLAN
```

```
-----  
Unique  (cost=72.35..94.91 rows=200 width=138) (actual time=0.272..0.475 rows=5 loops=1)  
-> Merge Join (cost=72.35..92.03 rows=1152 width=138) (actual time=0.263..0.398 rows=7 loops=1)  
    Merge Cond: ((r."char")::text = (s."char")::text)  
    -> Sort (cost=36.18..37.38 rows=480 width=138) (actual time=0.131..0.153 rows=10 loops=1)  
        Sort Key: r."char"  
        Sort Method: quicksort Memory: 17kB  
        -> Seq Scan on r (cost=0.00..14.80 rows=480 width=138) (actual time=0.012..0.035 rows=10 loops=1)  
    -> Sort (cost=36.18..37.38 rows=480 width=138) (actual time=0.108..0.135 rows=11 loops=1)  
        Sort Key: s."char"  
        Sort Method: quicksort Memory: 17kB  
        -> Seq Scan on s (cost=0.00..14.80 rows=480 width=138) (actual time=0.006..0.031 rows=10 loops=1)  
Total runtime: 0.589 ms  
(12 lignes)
```

