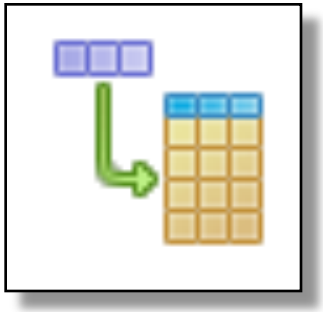
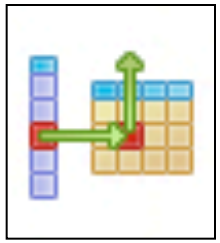
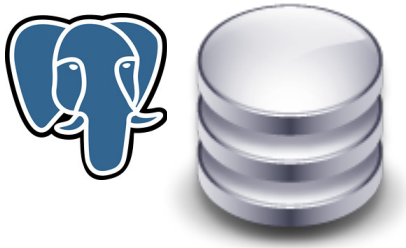
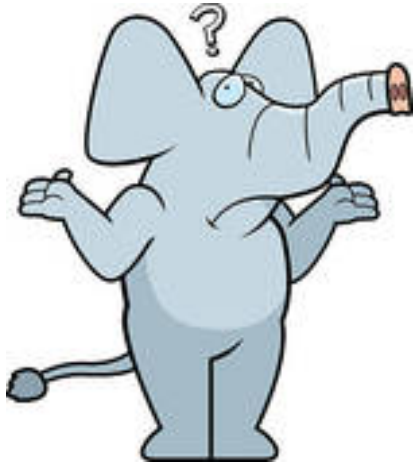






Conception Avancée de Bases de Données



PostGresSQL

PGadminIII

Explain

Traduction en cours

PostgreSQL PGadminIII Physical Query Plan

Query - stones sur postgres@localhost : 5432 *

Fichier Édition Requêtes Favoris Macros Affichage Aide

stones sur postgres@localhost : 5432

Éditeur SQL | Constructeur graphique de requêtes

```
SELECT
  auteurs.prenom,
  auteurs.nom,
  albums.album,
  albums.annee
FROM
  public.albums,
  public.auteurs,
  public.stones
WHERE
  albums.album_id = stones.album_id AND
  stones.auteurs_id = auteurs.auteurs_id;
```

Bloc notes

Panneau sortie

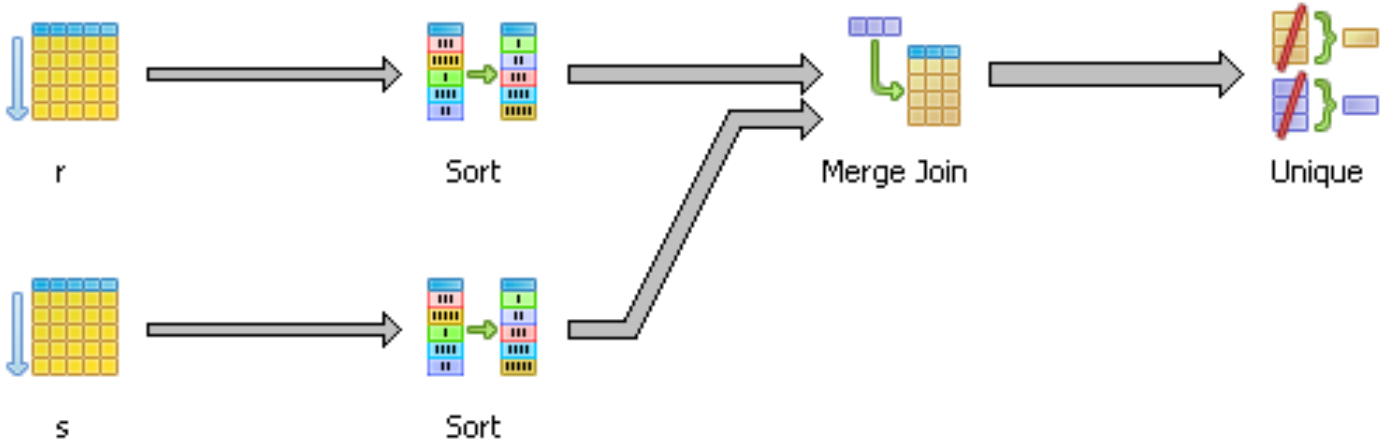
Sortie de données | Expliquer (Explain) | Messages | Historique

OK. | Unix | Ligne 13 Col 1 Caract. 224 | 10 lignes. | 16 ms

Explain Graphique

Panneau sortie

Sortie de données | Expliquer (Explain) | Messages | Historique



PostgreSQL Physical Query Plan

Query - stones sur postgres@localhost : 5432 *

Fichier Édition Requêtes Favoris Macros Affichage Aide

stones sur postgres@localhost : 5432

Éditeur SQL | Constructeur graphique de requêtes

```
SELECT
  auteurs.prenom,
  auteurs.nom,
  albums.album,
  albums.annee
FROM
  public.albums,
  public.auteurs,
  public.stones
WHERE
  albums.album_id = stones.album_id AND
  stones.auteurs_id = auteurs.auteurs_id;
```

Bloc notes

Panneau sortie

Sortie de données | Expliquer (Explain) | Messages | Historique

	QUERY PLAN text
1	Hash Join (cost=2.14..3.40 rows=7 width=38) (actual time=0.146..0.240 rows=7 loops=1)
2	Output: auteurs.prenom, auteurs.nom, albums.album, albums.annee
3	Hash Cond: (stones.auteurs_id = auteurs.auteurs_id)
4	-> Hash Join (cost=1.07..2.23 rows=7 width=26) (actual time=0.070..0.126 rows=7 loops=1)
5	Output: albums.album, albums.annee, stones.auteurs_id
6	Hash Cond: (stones.album_id = albums.album_id)
7	-> Seq Scan on stones (cost=0.00..1.07 rows=7 width=8) (actual time=0.006..0.022 rows=7 loops=1)
8	Output: stones.titre, stones.groupe, stones.auteurs_id, stones.album_id
9	-> Hash (cost=1.03..1.03 rows=3 width=26) (actual time=0.030..0.030 rows=3 loops=1)
10	Output: albums.album, albums.annee, albums.album_id
11	-> Seq Scan on albums (cost=0.00..1.03 rows=3 width=26) (actual time=0.005..0.013 rows=3 loops=1)
12	Output: albums.album, albums.annee, albums.album_id
13	-> Hash (cost=1.03..1.03 rows=3 width=20) (actual time=0.042..0.042 rows=3 loops=1)
14	Output: auteurs.prenom, auteurs.nom, auteurs.auteurs_id
15	-> Seq Scan on auteurs (cost=0.00..1.03 rows=3 width=20) (actual time=0.014..0.023 rows=3 loops=1)
16	Output: auteurs.prenom, auteurs.nom, auteurs.auteurs_id
17	Total runtime: 0.409 ms

OK. | Unix | Ligne 13 Col 1 Caract. 224 | 17 lignes. | 16 ms

PostgresQL Physical Query Plan

Query - EMP sur postgres@localhost : 5432 *

Fichier Édition Requêtes Favoris Macros Affichage Aide

EMP sur postgres@localhost : 5432

Éditeur SQL | Constructeur graphique de requêtes

```
SELECT EMP.NAME
FROM EMP,DEP,PROD
WHERE
EMP.DeptName = DEP.DeptName
AND EMP.EmpId = PROD.EmpId
AND DEP.Location = 'London'
AND PROD.ProdID = 'XHN2'
```

Bloc notes

Panneau sortie

Sortie de données | Expliquer (Explain) | Messages | Historique

emp

Hash Join

Nested Loop

dep

Hash

prod

Plan Node

Plan Node

OK. | Unix | Ligne 7 Col 25 Caract. 150 | 11 lignes. | 16 ms

PostgresQL Physical Query Plan

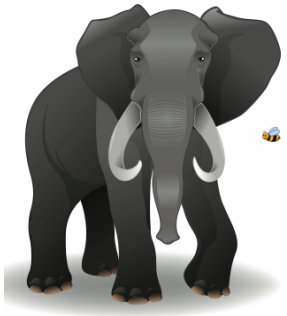


```
                                                    QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=11.76..36.52 rows=1 width=32) (actual time=0.249..0.333 rows=1 loops=1)
   Join Filter: ((emp.empid)::text = (prod.empid)::text)
   ->  Hash Join  (cost=11.76..24.25 rows=1 width=170) (actual time=0.120..0.181 rows=2 loops=1)
         Hash Cond: ((emp.deptname)::text = (dep.deptname)::text)
         ->  Seq Scan on emp  (cost=0.00..11.80 rows=180 width=308) (actual time=0.024..0.070 rows=16 loops=1)
         ->  Hash  (cost=11.75..11.75 rows=1 width=138) (actual time=0.032..0.032 rows=1 loops=1)
               ->  Seq Scan on dep  (cost=0.00..11.75 rows=1 width=138) (actual time=0.010..0.020 rows=1 loops=1)
                     Filter: ((location)::text = 'London'::text)
   ->  Seq Scan on prod  (cost=0.00..12.25 rows=1 width=138) (actual time=0.008..0.030 rows=5 loops=2)
         Filter: ((prod.prodid)::text = 'XHN2'::text)
 Total runtime: 0.469 ms
(11 lignes)
```

PostgresQL Physical Query Plan

```
                                                    QUERY PLAN
-----------------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=11.76..36.52 rows=1 width=32) (actual time=0.249..0.333 rows=1 loops=1)
   Join Filter: ((emp.empid)::text = (prod.empid)::text)
   ->  Hash Join  (cost=11.76..24.25 rows=1 width=170) (actual time=0.120..0.181 rows=2 loops=1)
         Hash Cond: ((emp.deptname)::text = (dep.deptname)::text)
         ->  Seq Scan on emp  (cost=0.00..11.80 rows=180 width=308) (actual time=0.024..0.070 rows=16 loops=1)
         ->  Hash  (cost=11.75..11.75 rows=1 width=138) (actual time=0.032..0.032 rows=1 loops=1)
               ->  Seq Scan on dep  (cost=0.00..11.75 rows=1 width=138) (actual time=0.010..0.020 rows=1 loops=1)
                     Filter: ((location)::text = 'London'::text)
   ->  Seq Scan on prod  (cost=0.00..12.25 rows=1 width=138) (actual time=0.008..0.030 rows=5 loops=2)
         Filter: ((prod.prodid)::text = 'XHN2'::text)
 Total runtime: 0.469 ms
(11 lignes)
```

Plan Node



PostgresQL Physical Query Plan

Query - EMP sur postgres@localhost : 5432 *

Fichier Édition Requêtes Favoris Macros Affichage Aide

EMP sur postgres@localhost : 5432

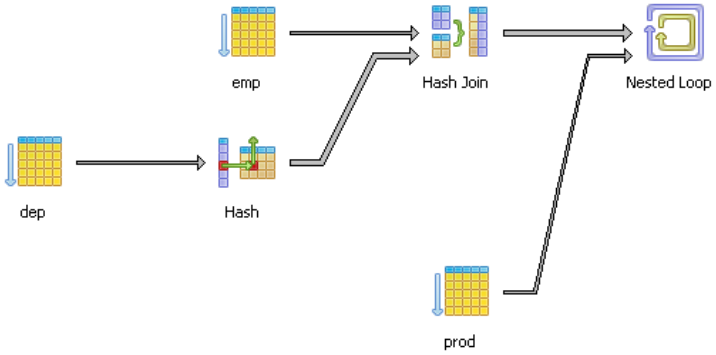
Éditeur SQL | Constructeur graphique de requêtes

```
SELECT EMP.NAME
FROM EMP,DEP,PROD
WHERE
EMP.DeptName = DEP.DeptName
AND EMP.EmpId = PROD.EmpId
AND DEP.Location = 'London'
AND PROD.ProdID = 'XHN2'
```

Bloc notes

Panneau sortie

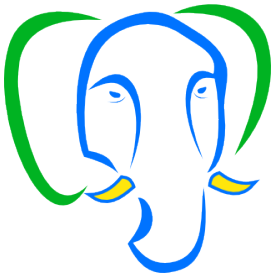
Sortie de données | Expliquer (Explain) | Messages | Historique

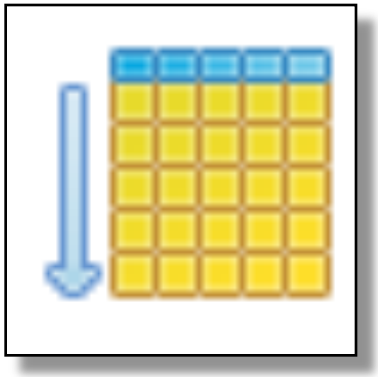


OK. | Unix | Ligne 7 Col 25 Caract. 150 | 11 lignes. | 16 ms

Select * from R

```
postgres=# explain select * from r;
                    QUERY PLAN
-----------------------------------------------------
 Seq Scan on r  (cost=0.00..14.80 rows=480 width=138)
(1 ligne)
```

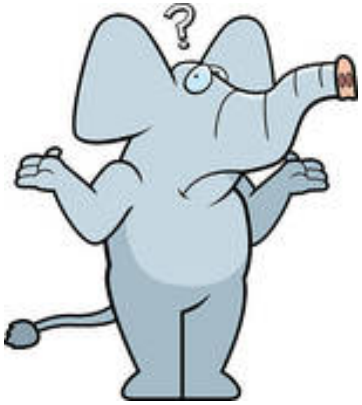


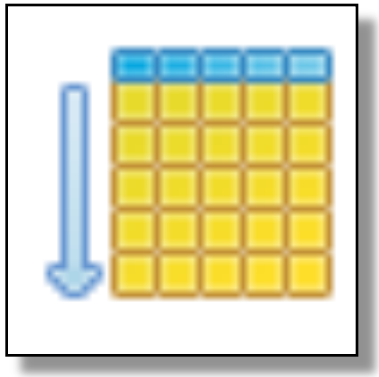


Eplain

- EXPLAIN prints the following information:
  - The type of operation required.
  - The estimated cost of execution.

```
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)
```

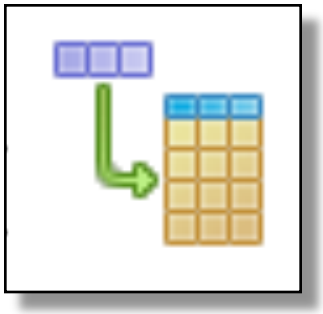




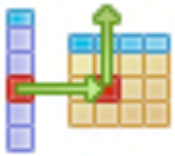
Plan Nodes

- Scans
  - Table scans (Sequential, Index, Bitmap, tid)
  - Other scans (Function, Values, Result)

- Joins
  - Nested Loop, Merge, Hash
- Set Operations, Partitioned Tables, and Inheritance
  - Append
  - SetOp Except, Intersect



- Miscellaneous
  - Sort, Aggregate, Unique, Limit
  - Materialize
  - SubPlan, Initplan



Sort

Query – emmanuelfuchs sur la socket locale *

Éditeur SQL | Constructeur graphique de requêtes

Requêtes précédentes | Supprimer | Tout supprimer

```
SELECT
  prod.prodid
FROM
  public.prod
ORDER BY
  prod.prodid ASC;
```

Panneau sortie

Sortie de données | Expliquer (Explain) | Messages | Historiqu

Unix | Ligne 7, Col 1, Caract. 71 | 3 lignes. | 18 ms

Query – cavbd sur postgres@localhost : 5432 *

cavbd sur postgres@localhost

Éditeur SQL | Constructeur graphique de requêtes

Requêtes précédentes | Supprimer | Tout supprimer

```
SELECT
  count(*)
FROM
  public.country;
```

Panneau sortie

Expliquer (Explain)

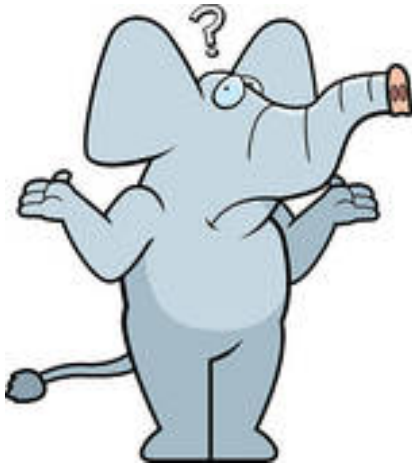
Sortie de données | Messages | Historique

	QUERY PLAN text
1	Aggregate (cost=7.99..8.00 rows=1 width=0)
2	-> Seq Scan on country (cost=0.00..7.39 rows=239 width=0)

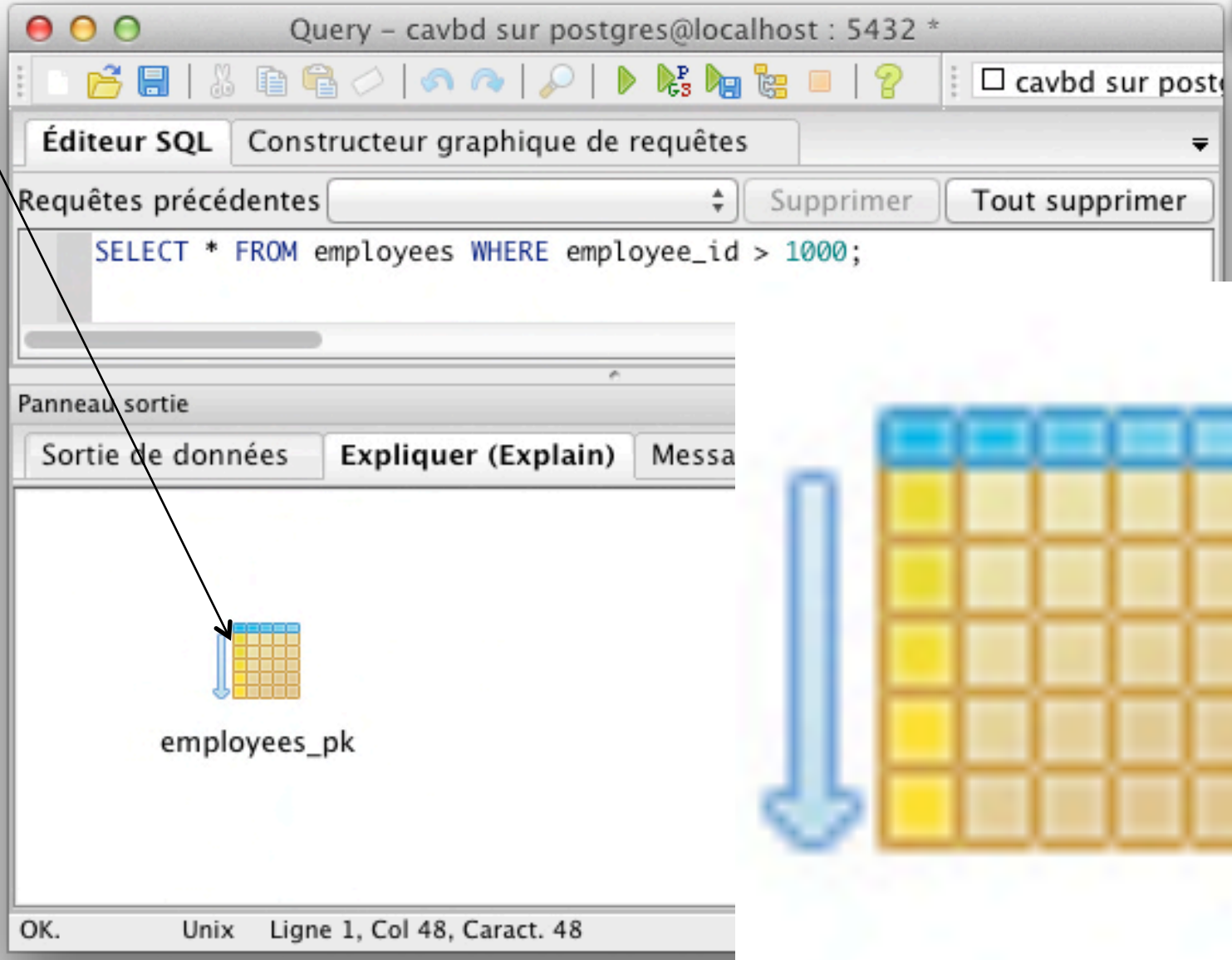
OK. Unix Ligne 4, Col 18, Caract. 44 2 lignes. 13 ms

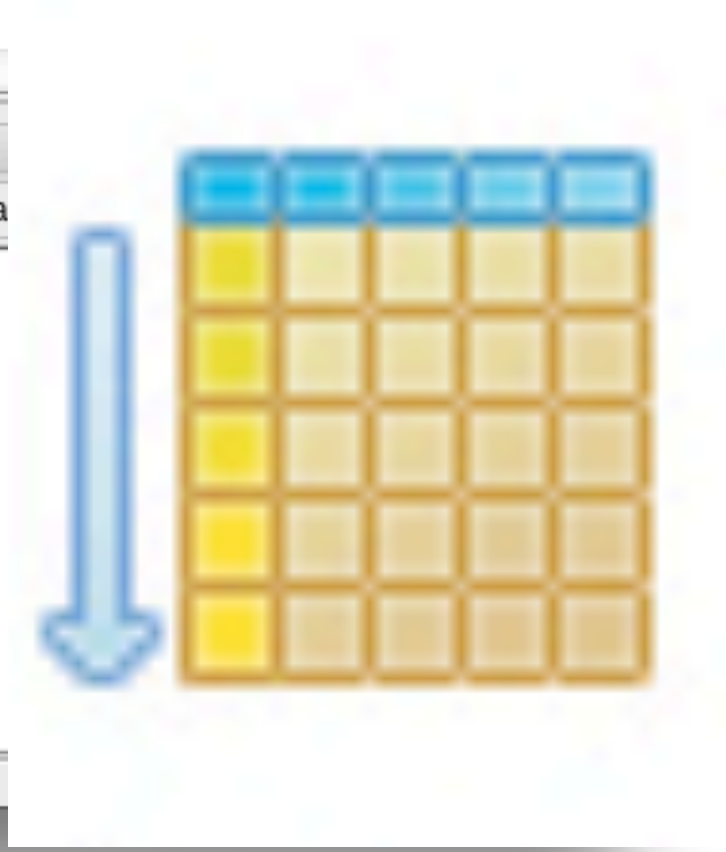
Aggregate functions

- *http://www.postgresql.org/docs/8.2/static/functions-aggregate.html*
- *Aggregate functions* compute a single result value from a set of input values.



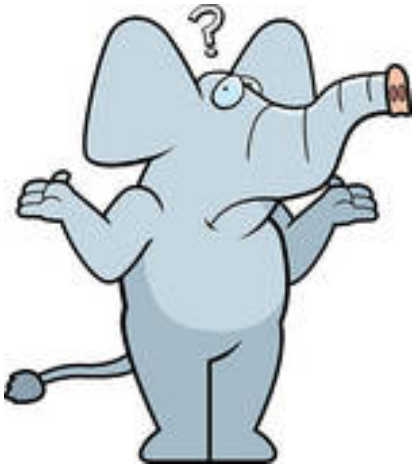
INDEX


Query – cavbd sur postgres@localhost : 5432 *
cavbd sur poste
Éditeur SQL
Constructeur graphique de requêtes
Requêtes précédentes
Supprimer
Tout supprimer
SELECT * FROM employees WHERE employee_id > 1000;
Panneau sortie
Sortie de données
Expliquer (Explain)
Messa
employees_pk
OK.
Unix
Ligne 1, Col 48, Caract. 48



EXPLAIN ANALYZE

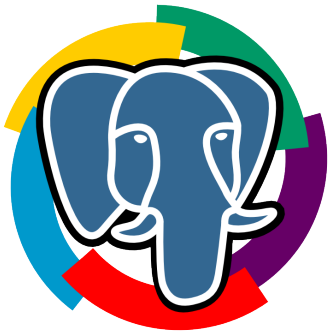
- If you specified EXPLAIN ANALYZE, prints the actual cost of execution.
- If you omit the ANALYZE keyword,
  - the query is planned but not executed,
  - and the actual cost is not displayed.



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



Query – cavbd sur postgres@localhost : 5432 *

cavbd sur postgres@localhost

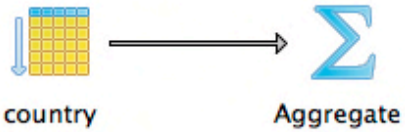
Éditeur SQL | Constructeur graphique de requêtes

Requêtes précédentes | Supprimer | Tout supprimer

```
SELECT
  avg(surfacearea)
FROM
  public.country;
```

Panneau sortie

Expliquer (Explain)



Sortie de données

	QUERY PLAN text
1	Aggregate (cost=7.99..8.00 rows=1 width=4)
2	-> Seq Scan on country (cost=0.00..7.39 rows=239 width=4)

OK. Unix Ligne 3, Col 6, Caract. 34 2 lignes. 14 ms

Sort

Query – emmanuelfuchs sur la socket locale *

Éditeur SQL | Constructeur graphique de requêtes

Requêtes précédentes | Supprimer | Tout supprimer

```
SELECT
  prod.prodid
FROM
  public.prod
ORDER BY
  prod.prodid ASC;
```

Panneau sortie

Sortie de données | Expliquer (Explain) | Messages | Historiqu

Unix | Ligne 7, Col 1, Caract. 71 | 3 lignes. | 18 ms

Sort

Query – emmanuelfuchs sur la socket locale *

Éditeur SQL | Constructeur graphique de requêtes

Requêtes précédentes | Supprimer | Tout supprimer

```
SELECT
  prod.prodid
FROM
  public.prod
ORDER BY
  prod.prodid ASC;
```

Panneau sortie

xpliquer (Explain) | Messages | Historiqu

Seq Scan
on prod
(cost=0.00..1.28 rows=28 width=5)

prod

Sort

Unix | Ligne 7, Col 1, Caract. 71 | 3 lignes. | 18 ms

Sort

Query – emmanuelfuchs sur la socket locale *

Éditeur SQL | Constructeur graphique de requêtes

Requêtes précédentes | Supprimer | Tout supprimer

```
SELECT
  prod.prodid
FROM
  public.prod
ORDER BY
  prod.prodid ASC;
```

Panneau sortie

Sortie de données | Ex | Historiqu

Sort

Sort Key: prodid
(cost=1.95..2.02 rows=28 width=5)

prod → Sort

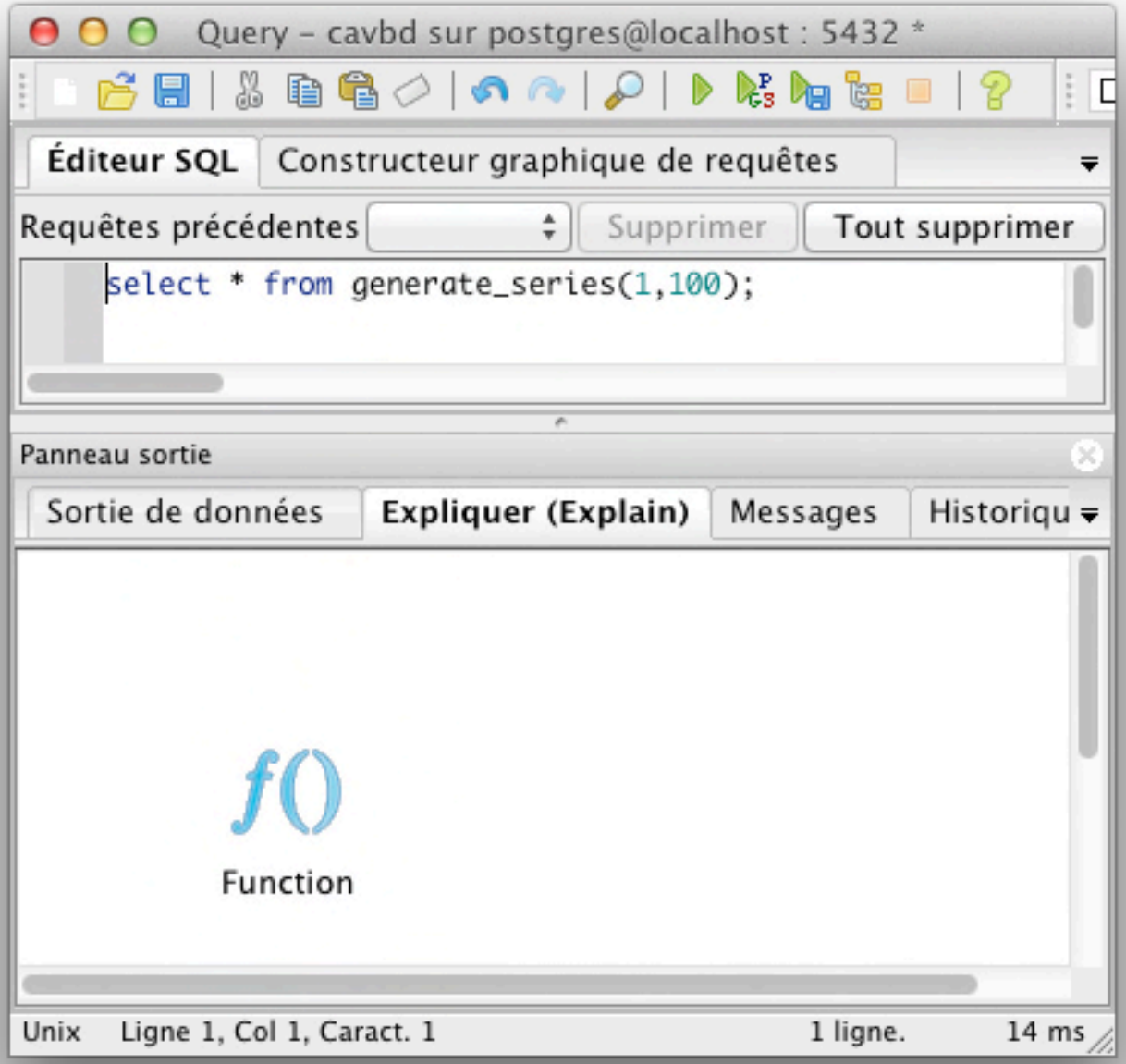
Unix | Ligne 7, Col 1, Caract. 71 | 3 lignes. | 18 ms

Exemples très simples

```
explain select * from generate_series(1,100);

explain values (1, 'un'), (2, 'deux'),(3, 'trois');

explain select 1;
```


Query – cavbd sur postgres@localhost : 5432 *
Éditeur SQL
Constructeur graphique de requêtes
Requêtes précédentes
Supprimer
Tout supprimer
select * from generate_series(1,100);
Panneau sortie
Sortie de données
Expliquer (Explain)
Messages
Historiqu
f()
Function
Unix
Ligne 1, Col 1, Caract. 1
1 ligne.
14 ms

Query – cavbd sur postgres@localhost : 5432 *

Éditeur SQL | Constructeur graphique de requêtes

Requêtes précédentes | Supprimer | Tout supprimer

```
select * from generate_series(1,100);
```

Panneau sortie

Sortie de données | Expliquer (Explain) | Messages | Historiqu

	generate_series integer
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	12
13	13
14	14
15	15
16	16
17	17
18	18
19	19
20	20
21	21
22	22

Unix Ligne 1, Col 1, Caract. 1 100 lignes. 17 ms

Query – cavbd sur postgres@localhost : 5432 *

Éditeur SQL | Constructeur graphique de requêtes

Requêtes précédentes | Supprimer | Tout supprimer

```
values (1,'un'), (2,'deux'),(3,'trois');
```

Panneau sortie

Sortie de données | Expliquer (Explain) | Messages | Historiqu

	column1 integer	column2 text
1	1	un
2	2	deux
3	3	trois

Unix Ligne 2, Col 1, Caract. 42 3 lignes. 185 ms

Query – cavbd sur postgres@localhost : 5432 *

Éditeur SQL | Constructeur graphique de requêtes

Requêtes précédentes | Supprimer | Tout supprimer

```
values (1,'un'), (2,'deux'),(3,'trois');
```

Panneau sortie

Sortie de données | Expliquer (Explain) | Messages | Historiqu

"*VALUES*"

Unix Ligne 2, Col 1, Caract. 42 1 ligne. 43 ms

explain values (1), (2),(3);

Query - postgres sur postgres@localhost : 5432 *

Fichier Édition Requêtes Favoris Macros Affichage Aide

postgres sur postgres@localhost : 5432

Éditeur SQL | Constructeur graphique de requêtes

```
VALUES (1, 'un'), (2, 'deux'), (3, 'trois');
```

Bloc notes

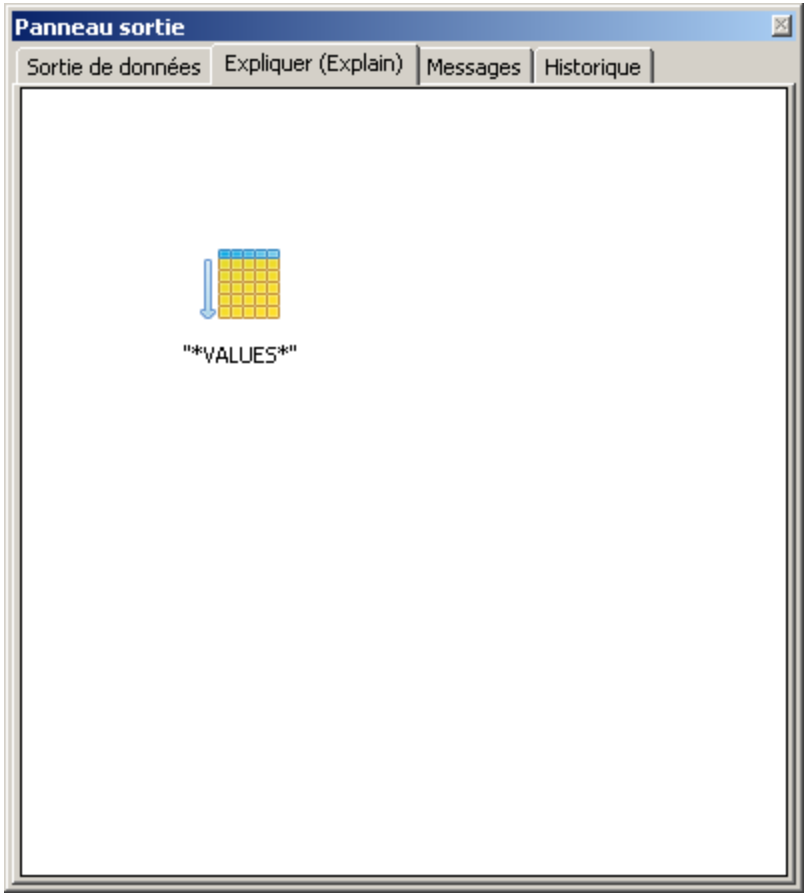
Panneau sortie

Sortie de données | Expliquer (Explain) | Messages | Historique

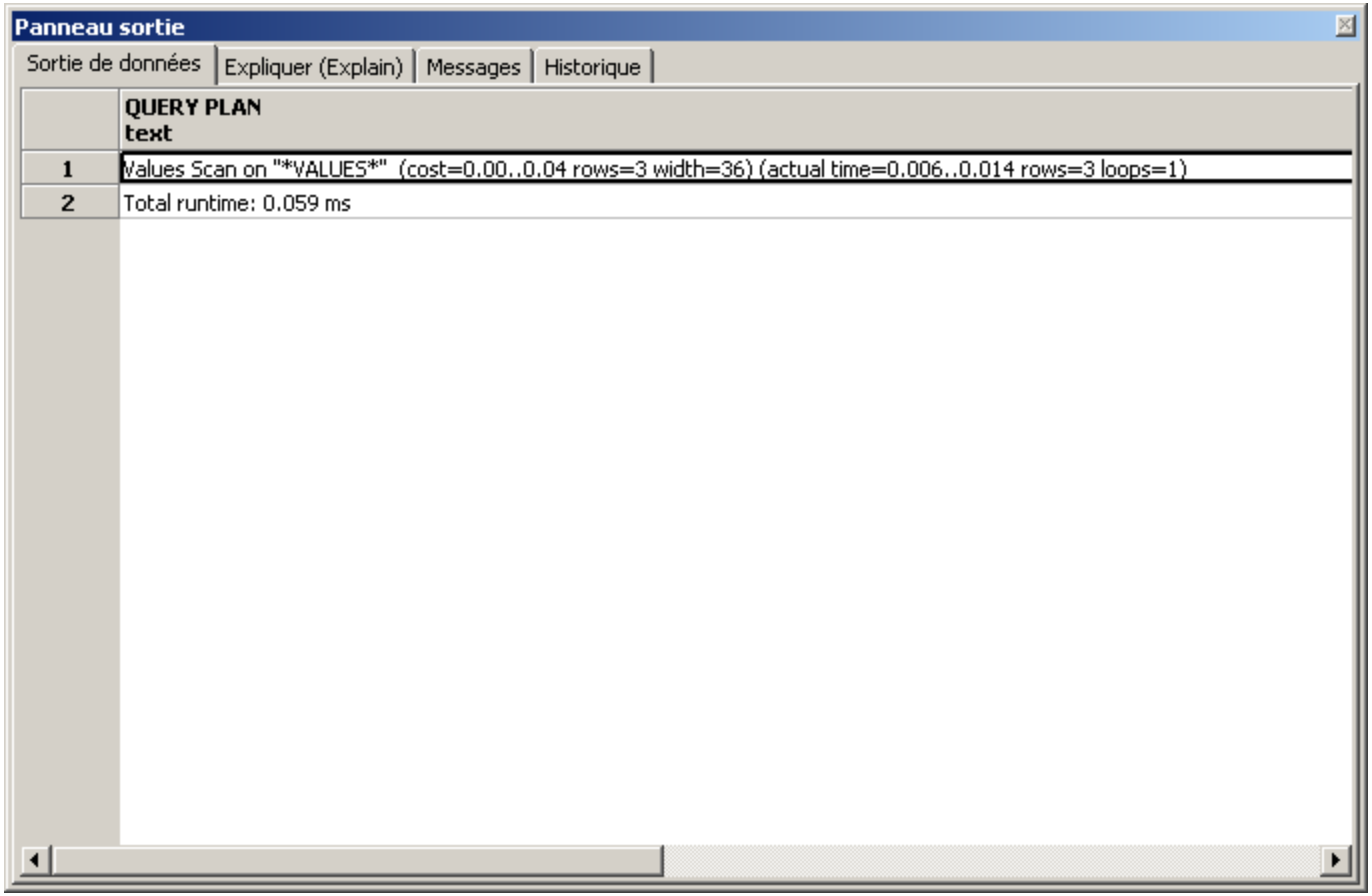
	column1 integer	column2 text
1	1	un
2	2	deux
3	3	trois

OK. | Unix | Ligne 1 Col 1 Caract. 1 | 3 lignes. | 16 ms

explain values (1), (2),(3);



explain values (1), (2),(3);



explain select 1;

Query - postgres sur postgres@localhost : 5432 *

Fichier Édition Requêtes Favoris Macros Affichage Aide

postgres sur postgres@localhost : 5432

Éditeur SQL | Constructeur graphique de requêtes

```
select 1;
```

Bloc notes

Panneau sortie

Sortie de données | Expliquer (Explain) | Messages | Historique

	?column? integer
1	1

OK. | Unix | Ligne 1 Col 1 Caract. 1 | 1 ligne. | 0 ms

explain select 1;

Query - postgres sur postgres@localhost : 5432 *

Fichier Édition Requêtes Favoris Macros Affichage Aide

postgres sur postgres@localhost : 5432

Éditeur SQL | Constructeur graphique de requêtes

```
select 1;
```

Bloc notes

Panneau sortie

Sortie de données | Expliquer (Explain) | Messages | Historique

Result

OK. | Unix | Ligne 1 Col 10 Caract. 10 | 2 lignes. | 16 ms

explain select 1;

Query - postgres sur postgres@localhost : 5432 *

Fichier Édition Requêtes Favoris Macros Affichage Aide

postgres sur postgres@localhost : 5432

Éditeur SQL | Constructeur graphique de requêtes

```
select 1;
```

Bloc notes

Panneau sortie

Sortie de données | Expliquer (Explain) | Messages | Historique

	QUERY PLAN text
1	Result (cost=0.00..0.01 rows=1 width=0) (actual time=0.003..0.005 rows=1 loops=1)
2	Total runtime: 0.032 ms

OK. | Unix | Ligne 1 Col 10 Caract. 10 | 2 lignes. | 16 ms

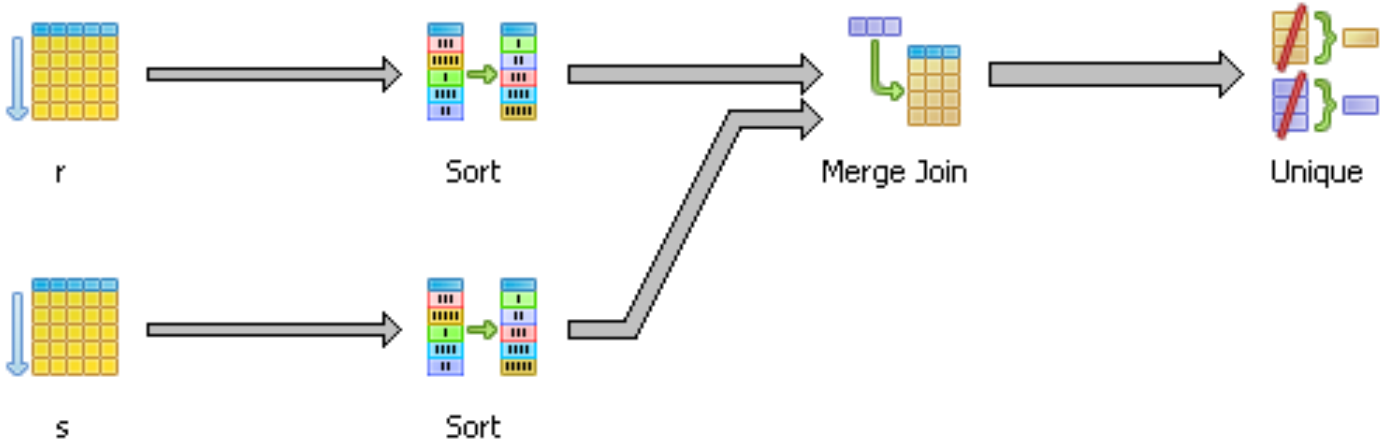
SELECT DISTINCT

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

Explain Graphique

Panneau sortie

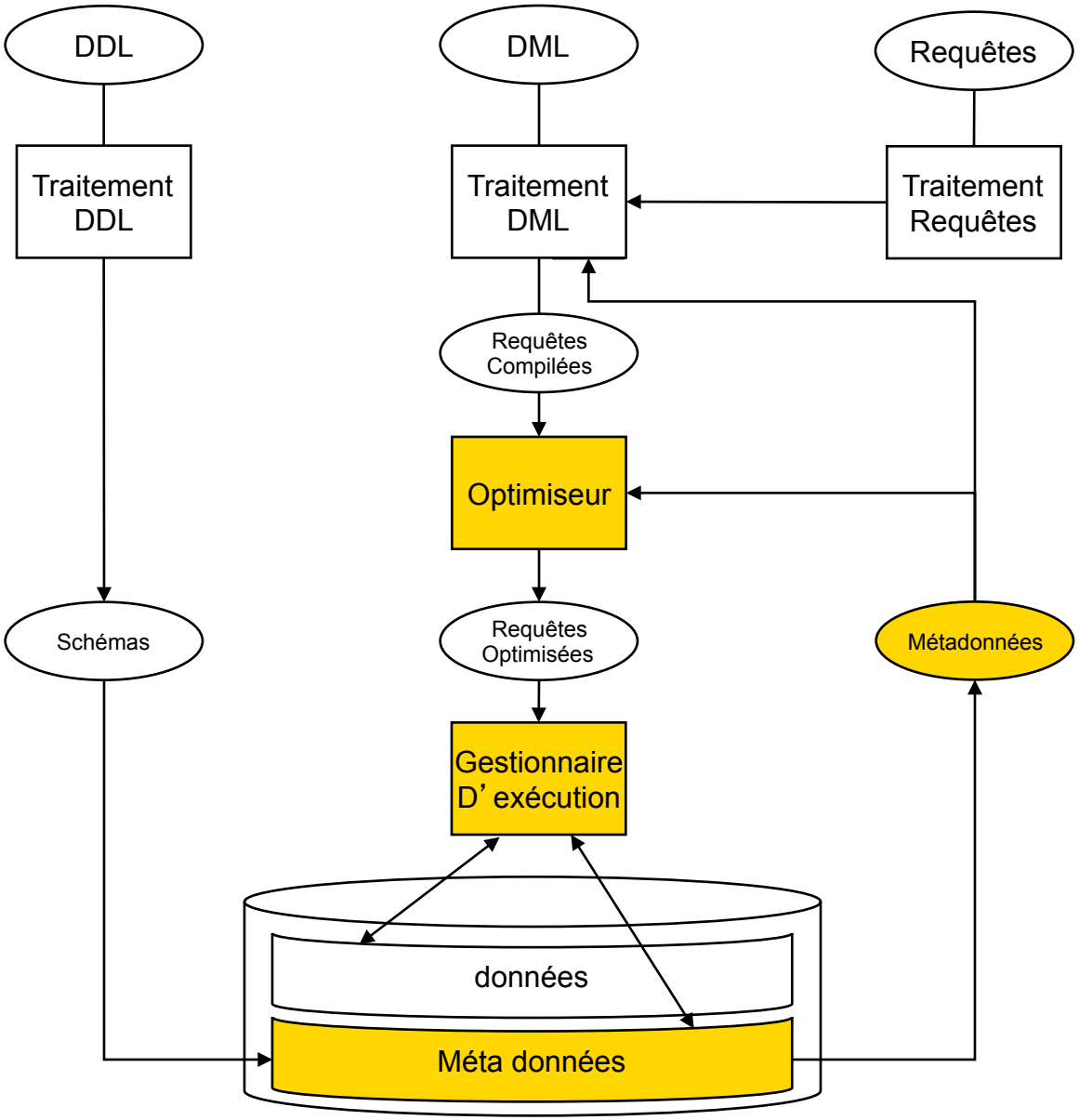
Sortie de données | Expliquer (Explain) | Messages | Historique



planner/optimizer

- The task of the *planner/optimizer* is to create an optimal execution plan
- The planner/optimizer starts by generating plans for scanning each individual relation (table) used in the query.

http://www.postgresql.org/docs/9.1/static/planner-optimizer.html



D' après C.J DATE

DDL : langage de définition des données; DML : langage de manipulation des données

Planner inputs