

# PEV2

PGSession - 21 Novembre 2019

Pierre Giraud



Il était une fois...

# EXPLAIN

Et ses plans d'exécution...

```

Limit (cost=1.27..3878.21 rows=5 width=172) (actual time=0.245..2.544 rows=5 loops=1)
-> Nested Loop (cost=1.27..48075.41 rows=62 width=172) (actual time=0.244..2.539 rows=5 l...
    -> Nested Loop (cost=0.84..2420.02 rows=65 width=85) (actual time=0.137..0.151 rows...
        -> Nested Loop (cost=0.42..2356.20 rows=5 width=85) (actual time=0.118..0.119...
            -> Seq Scan on top_challenge_list (cost=0.00..30.26 rows=58 width=57) (...
                Filter: (is_active AND ((template_challenge)::text = 'top_turnover'...
                Rows Removed by Filter: 26
            -> Index Scan using ref_people_xperf1 on ref_people (cost=0.42..40.09 r...
                Index Cond: (id_int = top_challenge_list.id_int_manager)
                Filter: (is_active AND (id_statut <> 2) AND (COALESCE(id_qualificat...
                Rows Removed by Filter: 4
        -> Index Scan using ref_genealogy_xperf5 on ref_genealogy (cost=0.42..12.56 r...
            Index Cond: (id_int = ref_people.id_int)
            Filter: is_active
    -> Index Scan using ref_people_xperf1 on ref_people filleuls (cost=0.42..0.82 rows=...
        Index Cond: (id_int = ref_genealogy.id_int_level)
        Filter: (is_active AND (id_type = ANY ('{5,14}'::integer[])))
        Rows Removed by Filter: 7
    SubPlan 1
        -> Aggregate (cost=361.46..361.47 rows=1 width=8) (actual time=0.233..0.233 rows=...
            -> Index Scan using ref_transaction_xperf4 on ref_transaction (cost=0.42..3...
                Index Cond: (id_int_agent_out = filleuls.id_int)
                Filter: (is_active AND (date_acte IS NOT NULL) AND (date_acte >= top_ch...
                Rows Removed by Filter: 100
    SubPlan 2
        -> Aggregate (cost=373.99..374.00 rows=1 width=8) (actual time=0.177..0.178 rows=...
            -> Index Scan using ref_transaction_xperf3 on ref_transaction ref_transactio...
                Index Cond: (id_int_agent_in = filleuls.id_int)
                Filter: (is_active AND (date_acte IS NOT NULL) AND (date_acte >= top_ch...
                Rows Removed by Filter: 102

```

Planning Time: 2.916 ms  
 Execution Time: 2.900 ms

Pas facile à lire, hein ?

# explain.depesz.com

explain.depesz.com

PostgreSQL's explain analyze made readable

[new explain](#)

[history](#)

[help](#)

[about](#)

[contact](#)

[login](#)

Result: W5zn : pike\_02Oct2019

[Settings](#)

[Add optimization](#)

#	exclusive	inclusive	rows x	rows	loops	node
1.	2.430	4,942,930.824	↓ 320.0	320	1	→ <a href="#">Sort</a> (cost=91,649.33..91,649.33 rows=1 width=3,399) (actual time=4,942,930.810..4,942,930.824 rows=320 loops=1) Sort Key: ((ui.lastname)::character varying(50)) COLLATE "en_US", ((ui.firstname "en_US", ui.duplicatename, ((login.loginname)::character varying(255)) COLLATE ((ex.trackingnumber)::character varying(50)) COLLATE "en_US", ex.expensedate varying(255)) COLLATE "en_US", ((ee.description)::character varying(255)) COLL ((expensetype3.name)::character varying(50)) COLLATE "en_US", ee.billtoclient, (**SELECT* 1".approverlastname)::character varying(50)) COLLATE "en_US", ((1".approverfirstname)::character varying(50)) COLLATE "en_US", elf.expensestat (**SELECT* 1".approvalcomments)::text) COLLATE "en_US", ((pj_1.code)::chara "en_US", ((pj_1.name)::character varying(255)) COLLATE "en_US", ((tk.code)::ch "en_US
2.						Initplan (forSort)
3.	0.003	0.003	↑ 1.0	1	1	→ <a href="#">Seq Scan</a> on systeminformation (cost=0.00..1.01 rows=1 width=4) (actual tin Buffers: shared hit=1
4.	0.005	0.005	↑ 1.0	1	1	→ <a href="#">Seq Scan</a> on systeminformation systeminformation_1 (cost=0.00..1.01 rows= (actual time=0.005..0.005 rows=1 loops=1) Buffers: shared hit=1
5.	6.396	4,942,928.386	↓ 320.0	320	1	→ <a href="#">Nested Loop Semi Join</a> (cost=6.72..91,647.30 rows=1 width=3,399) (actual time=188.986..4,942,928.386 rows=320 loops=1) Join Filter: (ex.userid = userlocation9.userid) Buffers: shared hit=2967970061
6.	4.026	4,942,912.673	↓ 1,331.0	1,331	1	→ <a href="#">Nested Loop Left Join</a> (cost=6.44..91,646.92 rows=1 width=393) (actual time=188.949..4,942,912.673 rows=1,331 loops=1)



# explain.depesz.com

- Existe depuis longtemps,
- Largement reconnu et utilisé,
- Maintenu (développement actif),
- Pratique et simple.

# explain.depesz.com

Seulement voilà...

- Pas très sexy,
- Affichage limité,
- Pas intégrable,
- Sur le web.

# PEV

## POSTGRES EXPLAIN VISUALIZER

[tatiyants.com/pev](https://tatiyants.com/pev)

# tatiyants.com/pev

- Plébicité (github 2.3k ★),
- Sexy, moderne,
- Mise en évidence des informations,
- Pas de stockage.

# tatiyants.com/pev

Seulement voilà...

- Abandonné,
- Pas intégrable.



# Les besoins

- Intégration dans d'autres outils (*TemBoard, DebugToolbar*)
- Mise à jour (versions de PostgreSQL)
- Ajout fonctionnalités

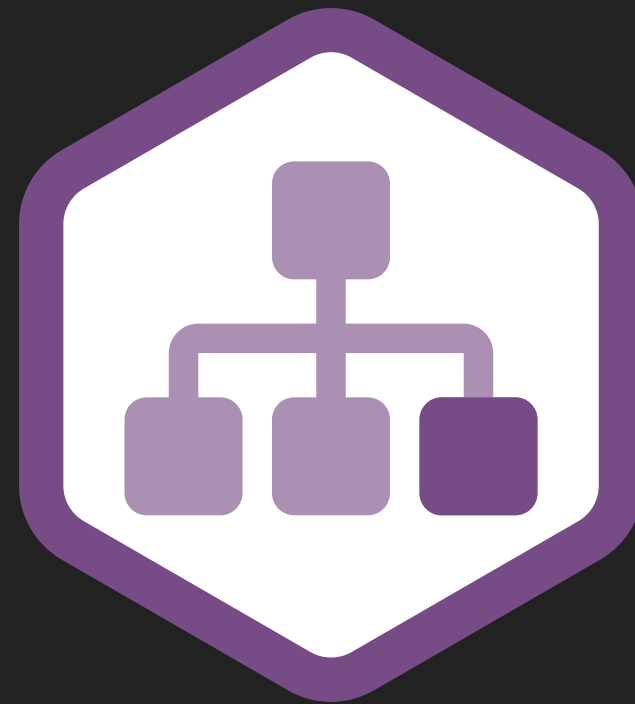
# Décision

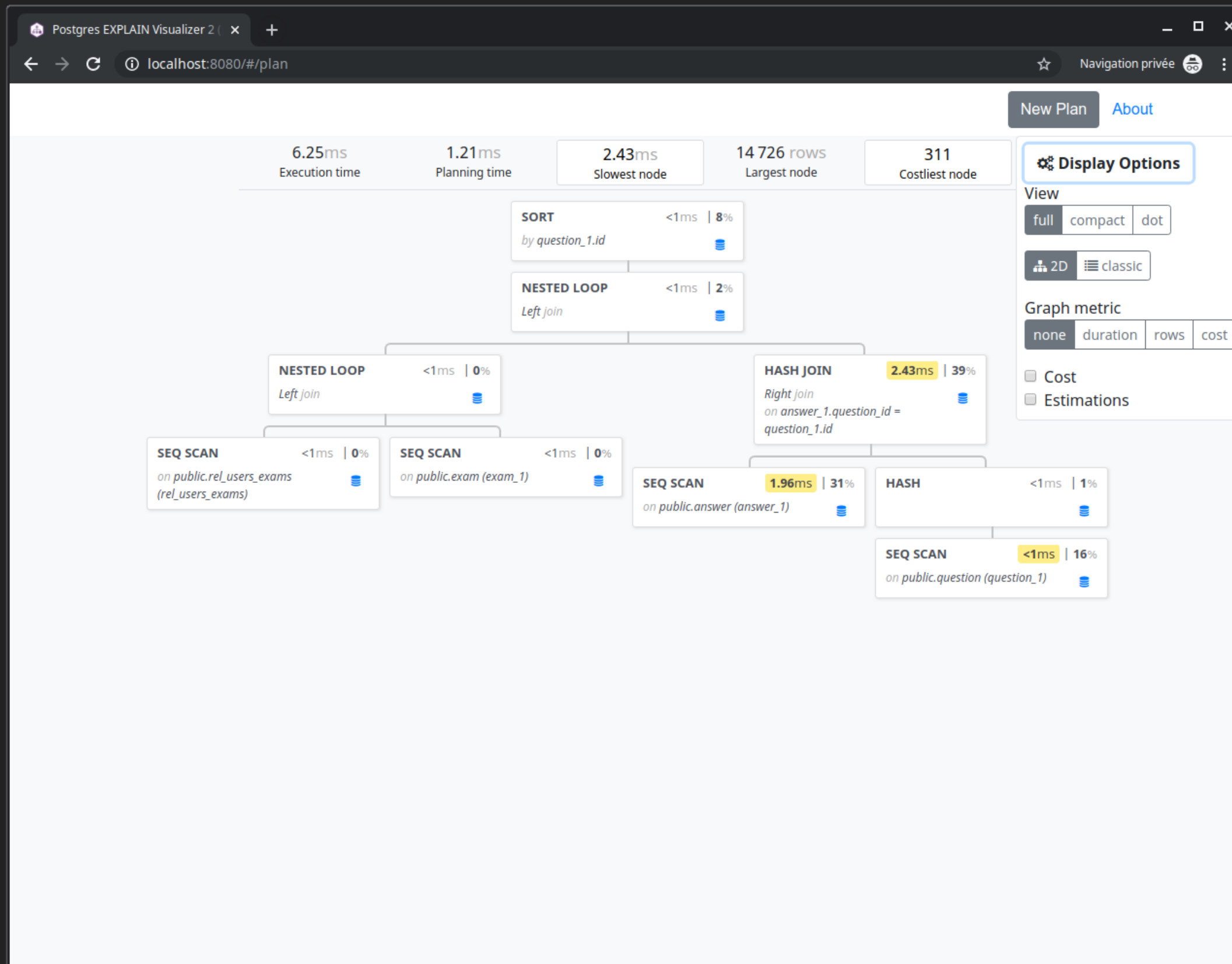
POC réécriture de PEV

→ très rapide car code bien écrit



# Please Welcome PEV2!





# PEV2

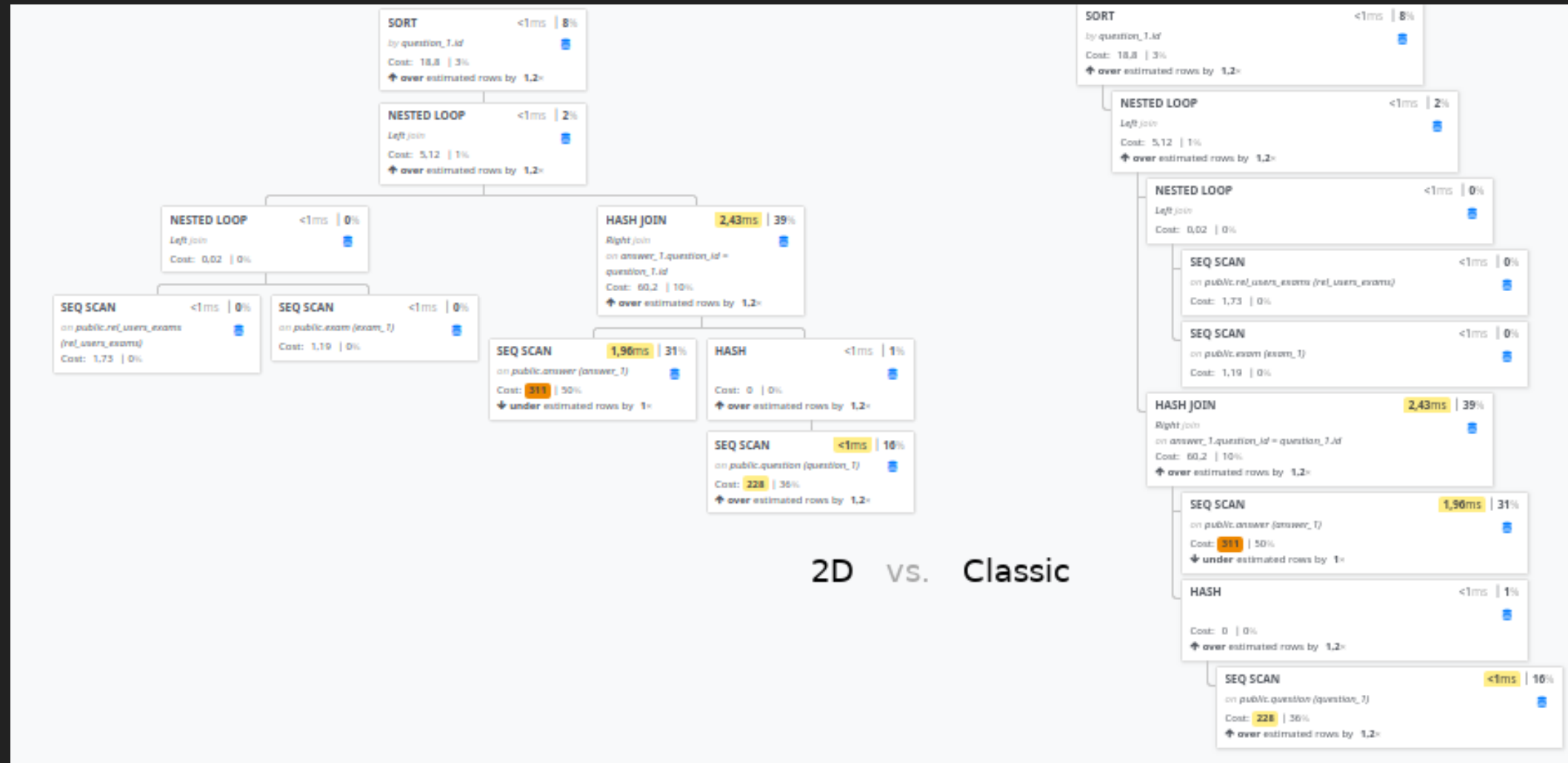
## NOUVELLES FONCTIONNALITÉS & DIFFÉRENCES

# TEXT PARSING

*“ Si c’est pas capable de lire un plan au format TEXT, j’m’en servirai pas. ”*

*Guillaume L.*

# VIEW OPTIONS



2D vs. Classic

# PARALLELISM

# MISE EN ÉVIDENCE

<b>SORT</b> by c.state,(sum(o.totalamount)) over estimated rows by 74x	1.36ms   0 %
<b>AGGREGATE</b> by c.state,cat.categoryname slowest under estimated rows by 1x	370.13ms   51 %
<b>HASH JOIN</b> Inner join on (o.orderid = ch.orderid) costliest largest under estimated rows by 1x	125.19ms   17 %

<b>SORT</b> by c.state, sum(o.totalamount) ↑ over estimated rows by 74x	1,36ms   0%
<b>AGGREGATE</b> by c.state, cat.categoryname Cost: 3 260   19% ↓ under estimated rows by 1x	370ms   51%
<b>HASH JOIN</b> Inner join on o.orderid = ch.orderid Cost: 5 830   34% ↓ under estimated rows by 1,2x	125ms   17%

*(PEV était limité aux superlatifs)*

# Utilisation comme composant



# DISPONIBLE SUR NPM

The screenshot shows the npm package page for 'pev2'. At the top, there is a search bar with the text 'Search packages' and a red 'Search' button. Below the search bar is a promotional banner for 'npm Orgs'. The package name 'pev2' is prominently displayed, followed by its version '0.1.15', status 'Public', and publication date 'Published 5 days ago'. A navigation bar contains links for 'Readme', 'Admin', '12 Dependencies', '0 Dependents', and '14 Versions'. The main content area includes a description: 'A VueJS component to show a graphical vizualization of a PostgreSQL execution plan.', a 'Greenkeeper enabled' badge, and a link to a 'Demo'. A 'Disclaimer' section states that the project is a rewrite of 'Postgres Explain Visualizer (pev)'. On the right side, there is an 'install' section with a terminal command '> npm i pev2', a 'weekly downloads' section showing '205' with a line graph, and a table with columns 'version' and 'license' showing '0.1.15' and 'none' respectively. At the bottom right, it indicates 'last publish 5 days ago'.

npm Search packages Search

Share private packages across your team with npm Orgs, now with simplified billing via the aws marketplace! [Learn more »](#)

**pev2**  
0.1.15 • Public • Published 5 days ago

[Readme](#) [Admin](#) [12 Dependencies](#) [0 Dependents](#) [14 Versions](#)

**pev2**

A VueJS component to show a graphical vizualization of a PostgreSQL execution plan.

Greenkeeper enabled

See [Demo](#).

**Disclaimer**

This project is a rewrite of the excellent [Postgres Explain Visualizer \(pev\)](#). Kudos go to [Alex Tatiyants](#).

install

```
> npm i pev2
```

± weekly downloads

205

version	license
0.1.15	none

last publish  
5 days ago

# INTÉGRATION

```
<div id="app">  
  <pev2 :plan-source="plan" :plan-query="query"></pev2>  
</div>
```

*(résumé)*

# INTÉGRATION

# INTÉGRATION

Application de démo dans le dépôt github.

[dalibo.github.io/pev2](https://dalibo.github.io/pev2)

*fonctionne dans le navigateur, pas de rétention*

# INTÉGRATION



**Dimitri Fontaine**  
@tapoueh

I'm publishing a [#PostgreSQL](#) Query Plan Visualizer at [theartofpostgresql.com//explain-plan-...](https://theartofpostgresql.com//explain-plan-...) ; that's an installation of PEV2! Give it a try, we have an experimental "save as PNG" feature!

À PARTAGER

[explain.dalibo.com](https://explain.dalibo.com)

Brought to you with ♥ by Dalibo ;-)

Title

Plan (text or JSON)

Paste execution plan  
Or drop a file

Query

Paste corresponding SQL query  
Or drop a file

Submit

For best results, use `EXPLAIN (ANALYZE, COSTS, VERBOSE, BUFFERS, FORMAT JSON)`  
`psql` users can export the plan to a file using `psql -qAt -f explain.sql > analyze.json`

Sample Plans ▾

+ New Plan

[explain.dalibo.com](https://explain.dalibo.com)

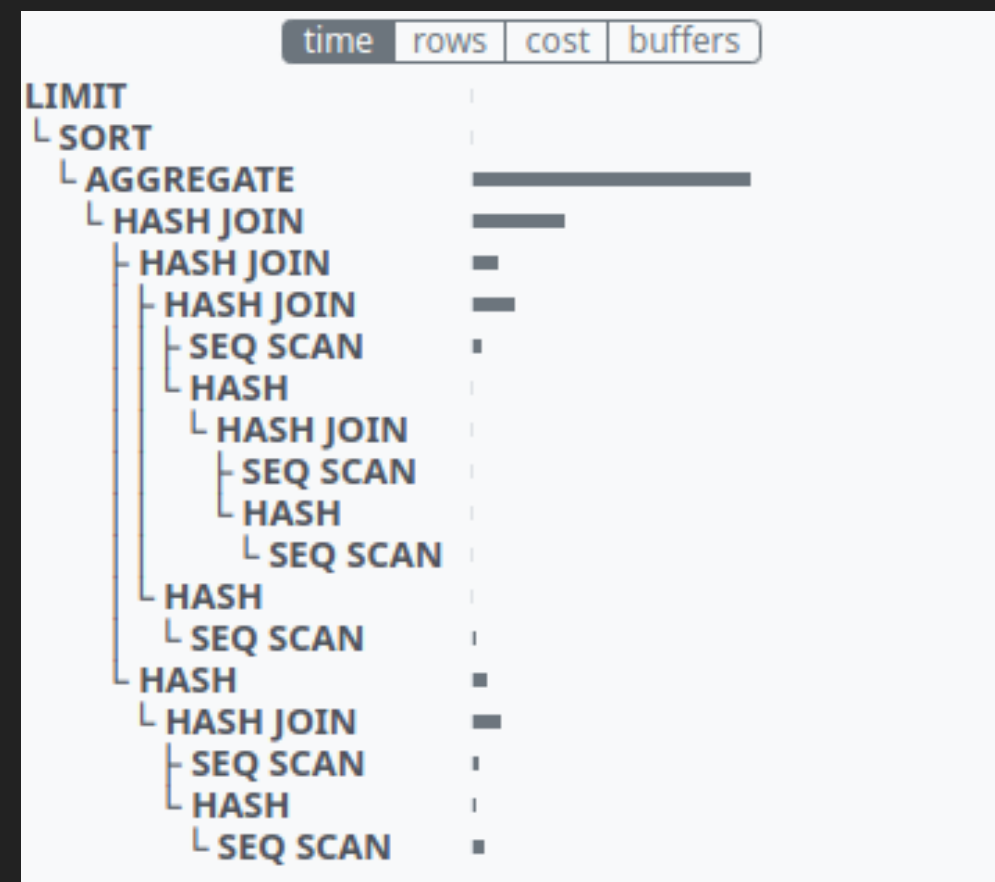
Vous pouvez envoyer vos plans en POST.

```
$ curl -Ls -w %{url_effective} -d '{"plan": "Result (cost=0.00..0.01 rows=1 width=32)"}'  
-H "Content-Type: application/json" -X POST https://explain.dalibo.com/new  
-o /dev/null | xargs xdg-open
```



# À VENIR

Diagramme (vue synthétique)



Feedbacks and contributions are welcome!

Pierre Giraud

DALIBO

 [PEV2 on github](#)

 [pgira](#)