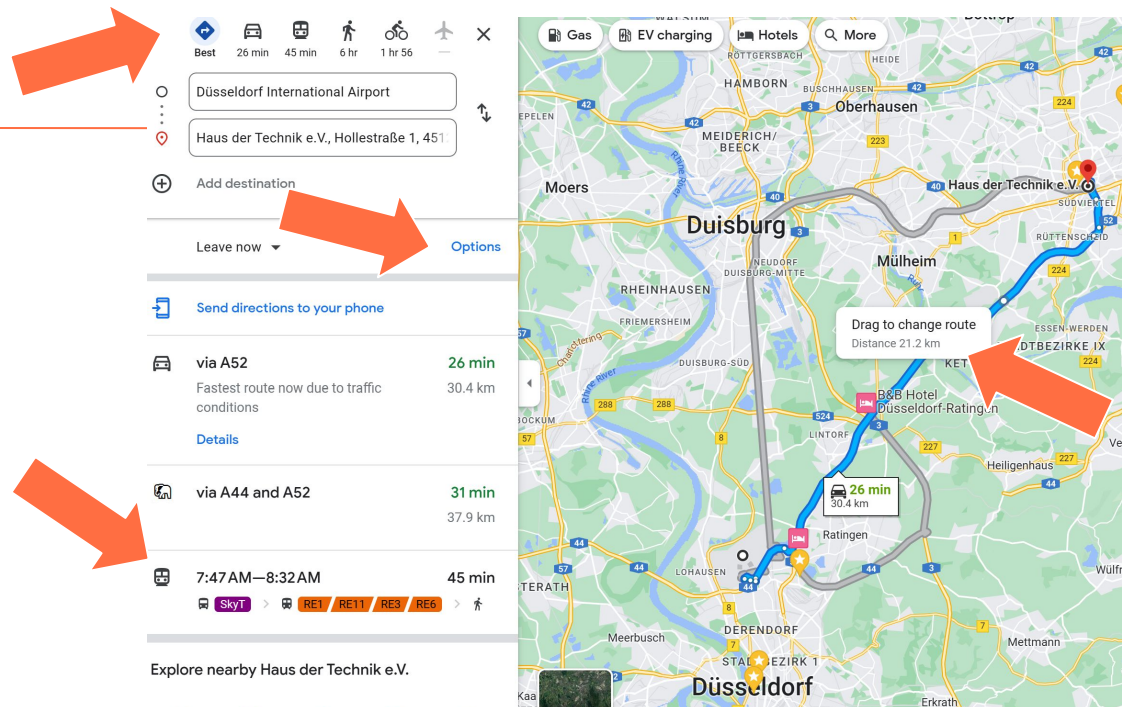# pg_hint_plan

get the right plan without surprises

Franck Pachot, Developer Advocate

Hints... why?

Would it make sense to provide Google Maps without the choice of travel mode, route options, drag to change route?

Hints... why in SQL?

SQL is a declarative language

the query planner generates the procedural code to access data

👉 You may want to understand its choices

👉 You may want to workaround bad choices

👉 You may know your data better, want stable plans...

Hints... how in PostgreSQL?

A harmless extension that has never been accepted in PG

Install [pg_hint_plan]( ) (🙏 NTT OSS)

```
FROM docker.io/postgres:14
ADD
https://github.com/ossc-db/pg_hint_plan/releases/download/
/REL14_1_4_0/pg_hint_plan14-1.4-1.el8.x86_64.rpm
RUN  apt-get update -y ; apt-get install -y alien ; alien
./pg_hint_plan*.rpm ; dpkg -i pg-hint-plan*.deb
```

yugabyteDB

Hints as directives in SQL comments

Because SQL is declarative, hints are not SQL -> comments

```
/*+
Leading ( (...) ) NestLoop(...) IndexScan(...)
Set(...) Rows(...) Parallel(...)
*/
select ... ; insert... ; prepare... ; explain ...
```
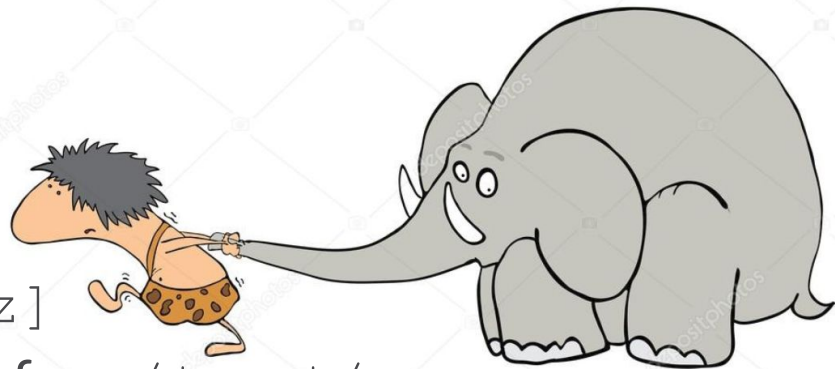
Easy, if you understand that you rarely need a single hint

Where to put /*+ ... */

At the beginning of a command

```
[0-9 \t\n,_()A-Za-z]
```
are the only characters allowed before /*+ ... */

- Syntax errors stop parsing, no nested comment, no --
- In the PREPARE, not the EXECUTE
- ⚠ with multi-statement commands ; ; ;   \; \; \;

in SQL   in psql
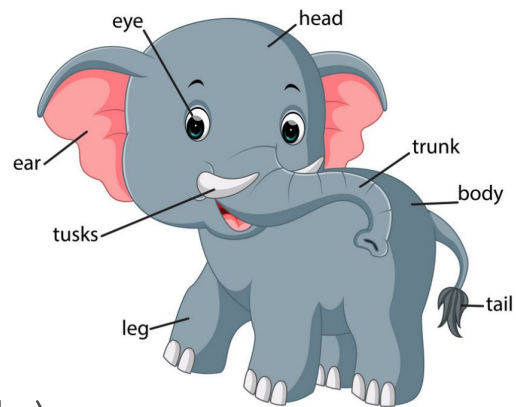
Hints reference tables and subqueries by their aliases

Many hints have a reference to tables
- by their alias (visible in execution plan)
- case sensitive (even with no quotes)
- Lists are not ordered:

```
HashJoin(a b c) = HashJoin(c a b)
```

- Nested Pairs are ordered:

```
Leading(  (a(b c)) ) != Leading(  ((a b)c)) )
```

Hints reference indexes by their name (be careful if you rename them!)

A bad name ignore all indexes

```
postgres=# /*+ IndexScan (accounts accounts_email) */ explain select * from accounts
           where user_id=7;
                                   QUERY PLAN
-------------------------------------------------------------------------------------
 Seq Scan on accounts  (cost=10000000000.00..10000000011.25 rows=1 width=520)


postgres=# /*+ IndexScan (accounts) */ explain select * from accounts where user_id=7;
                                   QUERY PLAN
-------------------------------------------------------------------------------------
 Index Scan using accounts_email_idx on accounts  (cost=0.14..8.16 rows=1 width=520)


postgres=# /*+ */ explain select * from accounts where user_id=7;
                                   QUERY PLAN
-------------------------------------------------------------------------------------
 Index Only Scan using accounts_email_idx on accounts  (cost=0.14..8.16 rows=1 width=520)
```
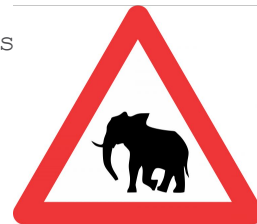
yugabyteDB

Troubleshooting: errors and log

By default:

```
INFO:  pg_hint_plan: hint syntax error
```

More info in the log (`on` or `verbose`):

```
set pg_hint_plan.debug_print=verbose;
```

To the client (`pg_hint_plan.message_level` defaults to log):

```
set client_min_messages = log;
```

What does a hint

Hints do not force anything

It can sets high cost for the unwanted access paths

Is evaluated during the query planning process

https://github.com/postgres/postgres/blob/master/src/backend/optimizer/path/costsize.c

```
131   Cost        disable_cost = 1.0e10;
```
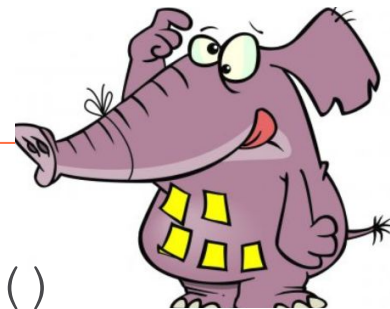
Demo:

Join order

Join direction

Scan method

Setting parameters

Cardinality correction

an hint on full hinting

For *n* table **aliases** in your (sub-)query

- *n*-1 nested pair of `(outer inner)` in `Leading()`
- for each pair: `NestLoop()`, `HashJoin()` or `MergeJoin()`
  they will have from 2 to *n* aliases (order doesn't matter)
- *n* scan method SeqScan(), IndexScan(), IndexOnlyScan()
  IndexScanRegexp(), ...

  https://github.com/ossc-db/pg_hint_plan#hints-list

🧮 count 6*n-2 closing (or opening) parentheses

Join selectivity estimation

You know you data better than PG

You can fix the cardinality

```
Rows( a b #42 )
```

or, better, apply a factor

```
Rows( a b c *0.3 )
```

Set parameters at query level

Example:

You know that Partition-wise join is good for one query but don't want to take more CPU and memory for other queries

```
/*+
  Set(enable_partitionwise_join true)
*/
```

no risk to forget to reset it back after
⚠️ for planning only, not execution

What if you cannot change the query?

```
create extension pg_hint_plan;

insert into hint_plan.hints
 (norm_query_string, application_name, hints) values (
 $sql$select * from table where a=$1 and b=?$sql$,
 'my_app', 'Leading( (a b) )'
);

set pg_hint_plan.enable_hint_table=on;
```

Applies hints to your application query
by matching the command text

- **$1,$2** are for prepared statements parameters
- **?** is for literals replaced before matching a query
- No final **;** except if there's one in your command

yugabyteDB

Hints in view?

Hints are ignored in views but applied in functions
  explain the view to get the aliases
  create a function on top of the view, with the hints

or create the function with the view text and a view on top of it


```
create or replace function myview()
returns setof myview as
$$
    /*+ NestLoop(demo1 demo2) */
    select * from myview;
$$ language sql;
```

Core message:

You may need hints, one day, maybe in emergency
Be sure it is installed and you know that they work

Helpful to experiment and learn about the query planner

fpachot@yugabyte.com
dev.to/FranckPachot
@FranckPachot