Evaluating and optimizing SQL query with Postgres Explain Analyze

An HTTP request journey

App	DB	View	Other
8%	61%	28%	4%

Get the query and Read the Analyze

What you need:

- A way to get the raw SQL query
- A way to execute the analyze in production and retrieve the result

Getting the raw sql

Django => .query

Sql Alchemy => .str()

SqlModel =>
print(query.compile(compile_kwargs={'literal_bindings': True))

- Query log
- Import pdb;
  pdb.set_trace()

Live example

An HTTP request journey

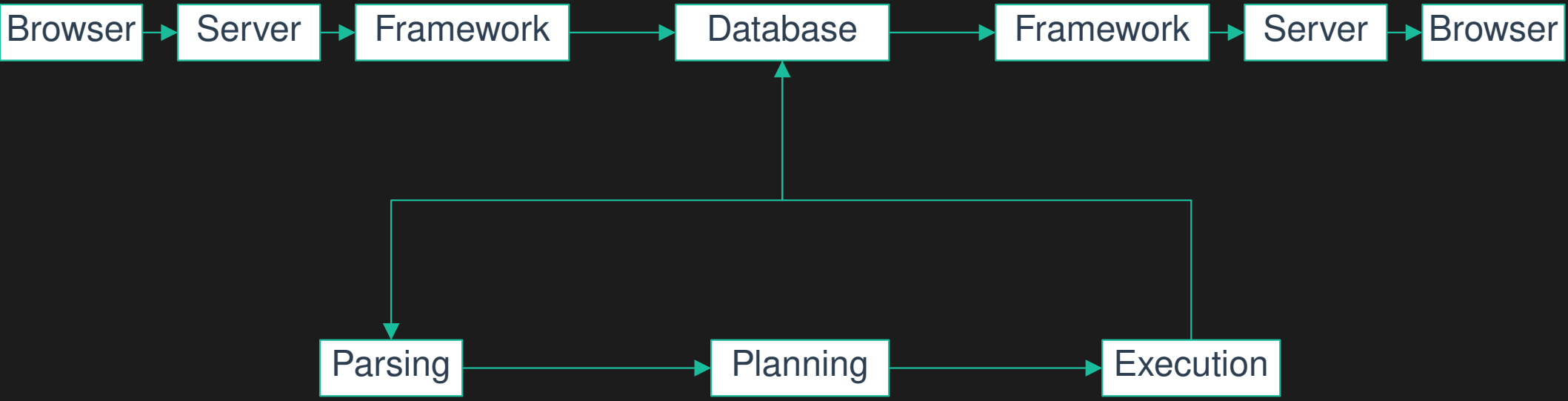

Table stats

```
afpy_db=# explain analyze select * from activities;
                                                          QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------
 Seq Scan on activities  (cost=0.00..114787.33 rows=6261033 width=56) (actual time=0.247..163.826 rows=6261121.00 loops=1)
   Buffers: shared hit=282 read=51895
 Planning Time: 0.051 ms
 Execution Time: 289.072 ms
(4 rows)
```

```
afpy_db=# select relpages, reltuples from pg_class where relname='activities';
 relpages |  reltuples
----------+--------------
    52177 | 6.261033e+06
(1 row)
```

```
afpy_db=# select 52177 * 1.0 + 6261033 * 0.01;
 ?column?
-----------
 114787.33
(1 row)
```

pg_stats

```
afpy_db=# select attname, null_frac, n_distinct, most_common_vals from pg_stats where tablename='activities' and attname in ('id', 'created_at', 'user_id', 'activity_type');
    attname    | null_frac | n_distinct |                          most_common_vals
---------------+-----------+------------+---------------------------------------------------------------------
 activity_type |         0 |          2 | {0,1}
 created_at    |         0 |         -1 |
 id            |         0 |         -1 |
 user_id       |         0 |       5006 | {3051,649,295,522,954,1367,2632,2990,3030,3098,4099,4747,4826,4939}
(4 rows)
```

```
    attname    |                                                  most_common_freqs

---------------+------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------
 activity_type | {0.5003333,0.49966666}
 created_at    |
 id            |
 user_id       | {0.00056666665,0.00053333334,0.00046666668,0.00046666668,0.00046666668,0.00046666668,0.00046666668,0.00046666668,0.00046666668,0.00046666668,0.00046666668,0.00046666668,0.000
46666668,0.00046666668}
```

Ways to optimize for a Query

1) Optimize the raw query
2) Add an index or remove one
3) Upgrade your database spec: work_mem and shared buffers

Live example

From sequential scan to index only scan

Conclusion How to Automate the Process?

- HypoPG
- PG analyze

Questions