Monitoring PostgreSQL At Scale

@LukasFittl

@LukasFittl

citusdata

pganalyze

Statistics That Matter

Two Tables To Remember

Breaking Down High-Level Statistics

Log Events Worth Knowing

Fingerprinting & Tracing Queries

Statistics That Matter

Two Tables To Remember

Breaking Down High-Level Statistics

Log Events Worth Knowing

Fingerprinting & Tracing Queries

Postgres Statistics
Tables

1 “Block” = 8 kB

(usually, check block_size to confirm)

Tuple = Row

Statistics Are Often Counters

Counts only go up*,
calculate diffs!

* except when reset / overrun

@LukasFittl

Schema Statistics

pg_stat_user_tables

relname:	name of the table
seq_scan:	# of sequential scans
idx_scan:	# of index scans
n_tup_(ins/del/upd):	# of rows modified
n_live_tup:	live rows
n_dead_tup:	dead rows
last_(auto)vacuum:	last VACUUM
last_(auto)analyze:	last ANALYZE
…	

Index Hit Rate

```
SELECT relname, n_live_tup, seq_scan + idx_scan,
       100 * idx_scan / (seq_scan + idx_scan)
  FROM pg_stat_user_tables
 ORDER BY n_live_tup DESC
```

Target: >= 95% on large, active tables

@LukasFittl

pg_statio_user_tables

relname:	name of the table
heap_blks_read:	blocks from disk / OS cache
heap_blks_hit:	blocks from buffer cache
idx_blks_read:	index blks from disk
idx_blks_hit:	index blks from buffer cache
	…

Table Cache Hit Rate

```
SELECT sum(heap_blks_hit) /
  nullif(sum(heap_blks_hit + heap_blks_read),0)
  FROM pg_statio_user_tables
```

Target: >= 99%

Query Workload

@LukasFittl

pg_stat_activity

pid:	process ID
backend_type:	"client backend" vs internal processes
state:	idle/active/idle in transaction
state_change:	time of state change
query:	current/last running query
backend_start:	process start time
xact_start:	TX start time
query_start:	query start time
wait_event:	what backend is waiting for (e.g. Lock, I/O, etc)

…

@LukasFittl

# of Connections By State

```
SELECT state,
       backend_type,
       COUNT(*)
  FROM pg_stat_activity
 GROUP BY 1, 2
```

@LukasFittl

Longest Running Query

```
SELECT now() - query_start,
       query
  FROM pg_stat_activity
 WHERE state = ‘active’
 ORDER BY 1
 LIMIT 1
```

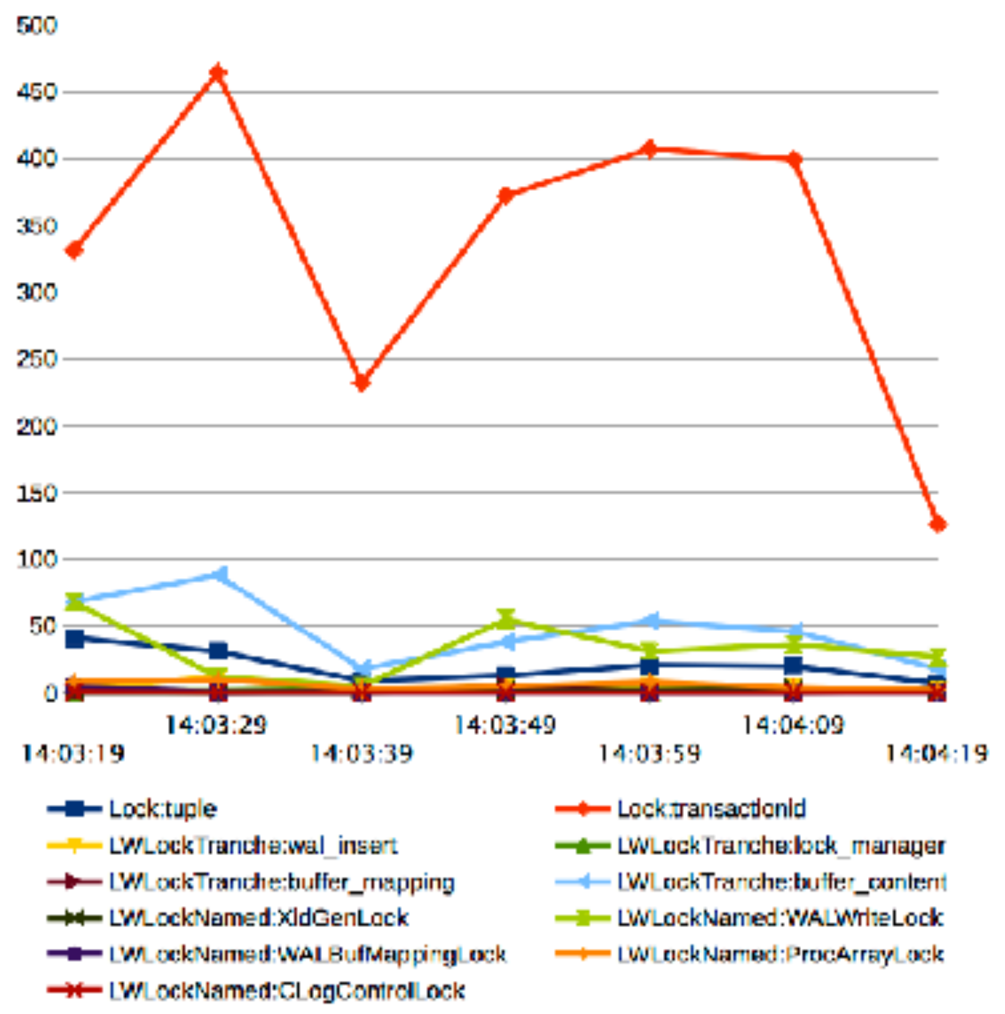
@LukasFittl

Age Of Oldest Transaction

```
SELECT MAX(now() - xact_start)
  FROM pg_stat_activity
 WHERE state <> ‘idle’
```

@LukasFittl

pg_stat_activity lock information



https://github.com/postgrespro/pg_wait_sampling

@LukasFittl

pg_stat_statements

1. Install postgresql contrib package (if not installed)
2. Enable in postgresql.conf

   ```
   shared_preload_libraries = ‘pg_stat_statements’
   ```
3. Restart your database
4. Create the extension

   ```
   CREATE EXTENSION pg_stat_statements;
   ```

@LukasFittl

pg_stat_statements

`SELECT * FROM pg_stat_statements;`

```
userid              | 10
dbid                | 1397527
query               | SELECT * FROM x WHERE
calls               | 5
total_time          | 15.249
rows                | 0
shared_blks_hit     | 451
shared_blks_read    | 41
shared_blks_dirtied | 26
shared_blks_written | 0
local_blks_hit      | 0
```

@LukasFittl

Supported on cloud platforms

amazon web services™

cituscloud

```
queryid      | 1720234670
query        | SELECT * FROM x WHERE y = ?
calls        | 5
total_time   | 15.249
```

Query + No. of Calls + Avg Time

@LukasFittl

Avg. Shared Buffer Hit Rate

```
shared_blks_hit       | 2447215
shared_blks_read      | 55335
```

```
hit_rate = shared_blks_hit /
           (shared_blks_hit + shared_blks_read)
```

97.78% Cache Hit Rate

Time spent reading/writing to disk

```
track_io_timing = on
```

```
blk_read_time        | 14.594
blk_write_time       | 465.661
```

@LukasFittl

pg_qtop

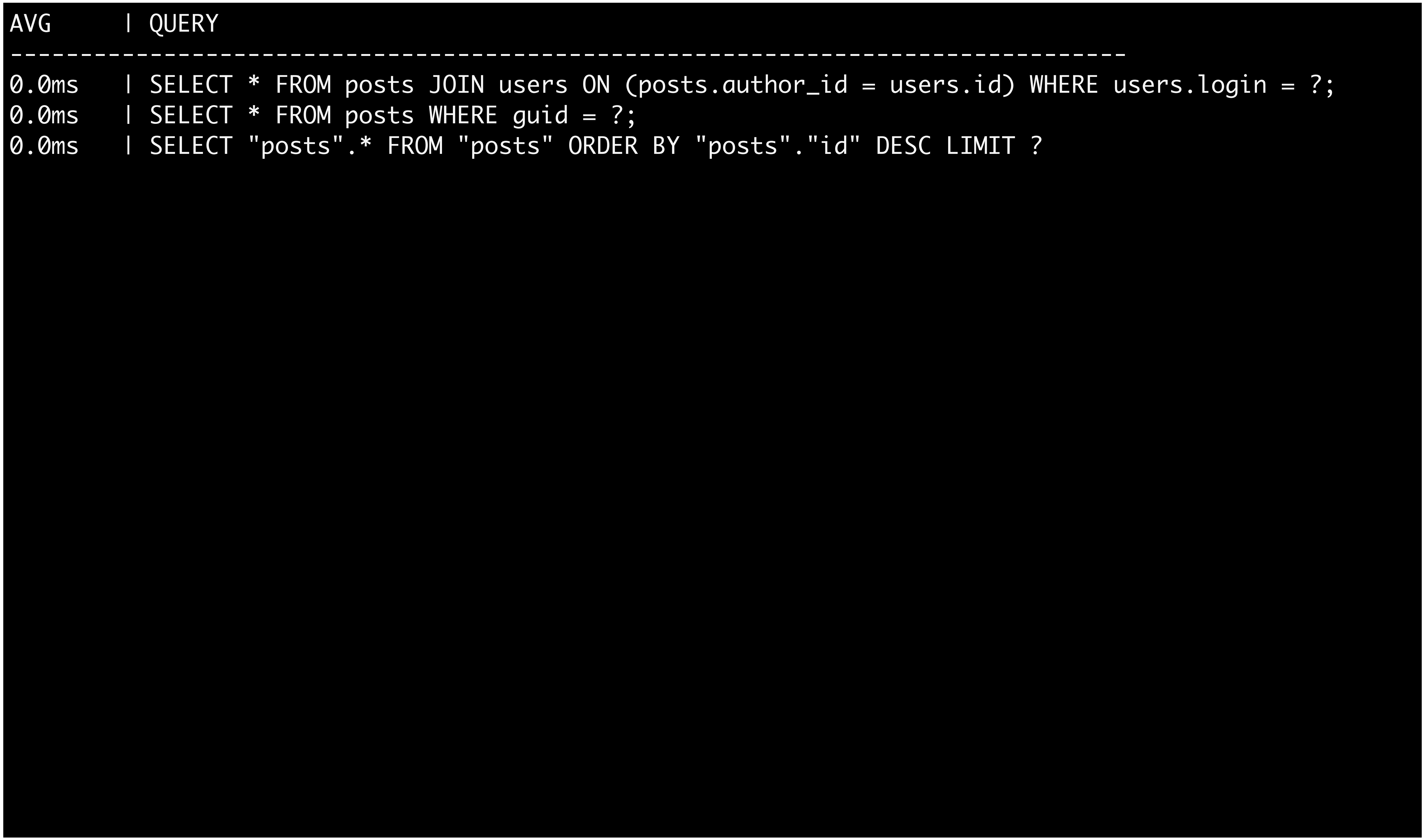
Simple top-like tool that shows pg_stat_statements data

https://github.com/lfittl/pg_qtop

pg_qtop -d testdb

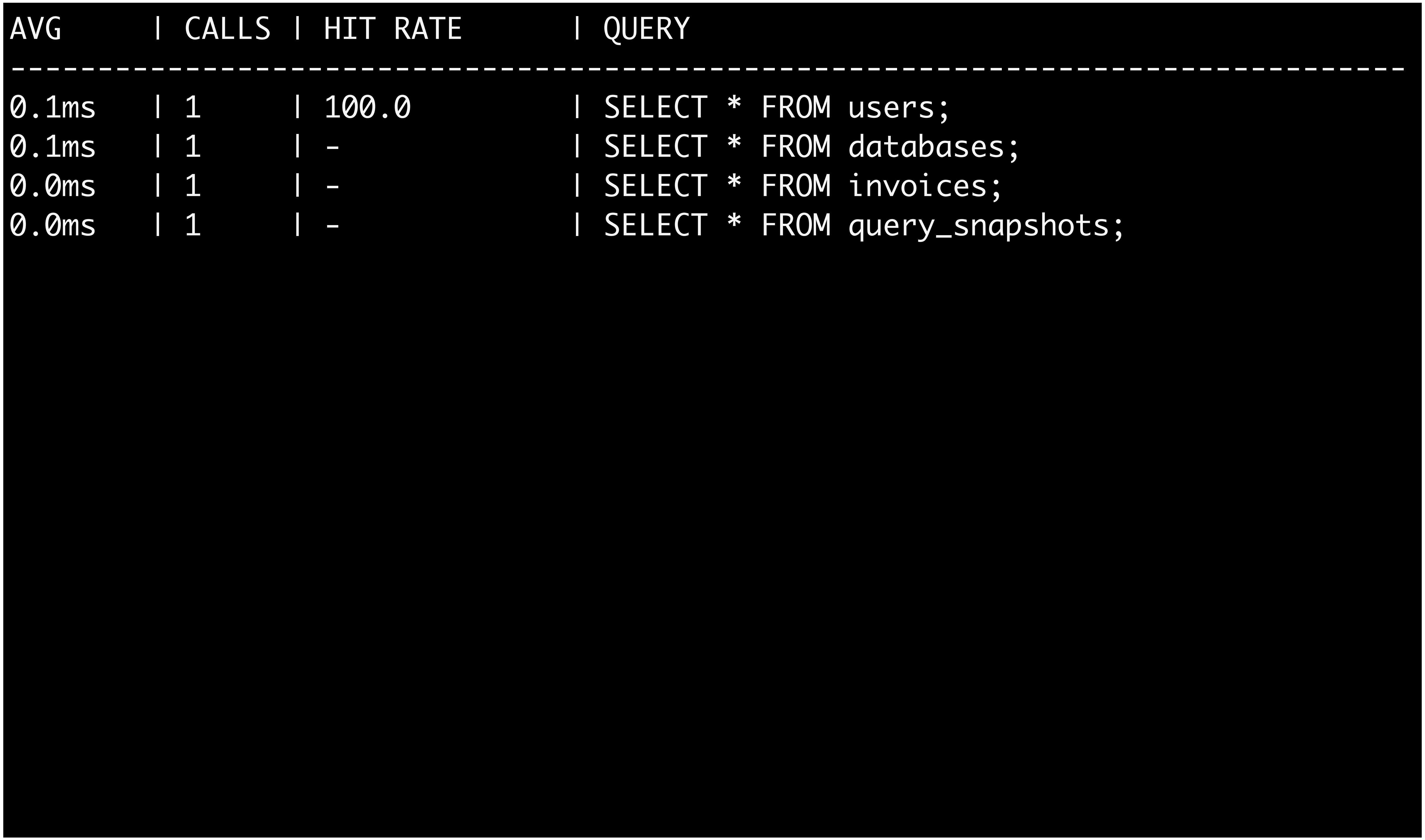
```
AVG     | QUERY
--------------------------------------------------------------------------------
10.7ms  | SELECT oid, typname, typelem, typdelim, typinput FROM pg_type
3.0ms   | SET time zone 'UTC'
0.4ms   | SELECT a.attname, format_type(a.atttypid, a.atttypmod), pg_get_expr(d.adbin, d.adrelid),
a.attnotnull, a.atttypid, a.atttypmod FROM pg_attribute a LEFT JOIN pg_attrdef d ON a.attrelid
= d.adrelid AND a.attnum = d.adnum WHERE a.attrelid = ?::regclass AND a.attnum > ? AND NOT
a.attisdropped ORDER BY a.attnum
0.2ms   | SELECT pg_stat_statements_reset()
0.1ms   | SELECT query, calls, total_time FROM pg_stat_statements
0.1ms   | SELECT attr.attname FROM pg_attribute attr INNER JOIN pg_constraint cons ON attr.attrelid
= cons.conrelid AND attr.attnum = cons.conkey[?] WHERE cons.contype = ? AND cons.conrelid = ?:
:regclass
0.0ms   | SELECT COUNT(*) FROM pg_class c LEFT JOIN pg_namespace n ON n.oid = c.relnamespace WHERE
c.relkind in (?,?) AND c.relname = ? AND n.nspname = ANY (current_schemas(?))
0.0ms   | SELECT * FROM posts JOIN users ON (posts.author_id = users.id) WHERE users.login = ?;
0.0ms   | SET client_min_messages TO 'panic'
0.0ms   | set client_encoding to 'UTF8'
0.0ms   | SHOW client_min_messages
0.0ms   | SELECT * FROM ad_reels WHERE id = ?;
0.0ms   | SELECT * FROM posts WHERE guid = ?;
0.0ms   | SELECT ?
0.0ms   | SET client_min_messages TO 'warning'
0.0ms   | SET standard_conforming_strings = on
0.0ms   | SELECT "posts".* FROM "posts" ORDER BY "posts"."id" DESC LIMIT ?
0.0ms   | SHOW TIME ZONE
```

pg_qtop -d testdb **-t posts**



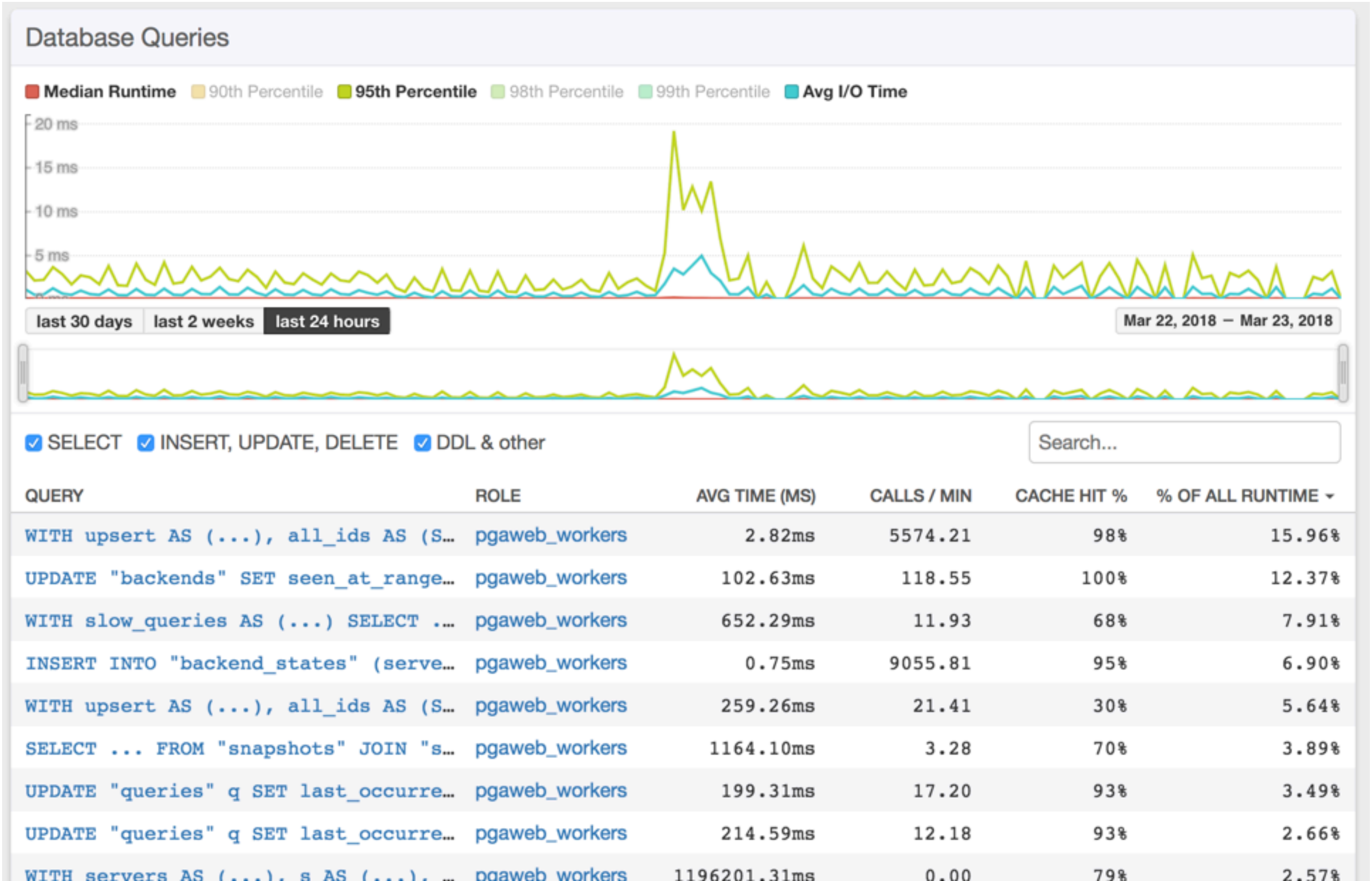
```
AVG     | QUERY
--------------------------------------------------------------------------------
0.0ms   | SELECT * FROM posts JOIN users ON (posts.author_id = users.id) WHERE users.login = ?;
0.0ms   | SELECT * FROM posts WHERE guid = ?;
0.0ms   | SELECT "posts".* FROM "posts" ORDER BY "posts"."id" DESC LIMIT ?
```

pg_qtop -d testdb **-s select**



```
AVG      | CALLS | HIT RATE        | QUERY
--------------------------------------------------------------------------------
0.1ms    | 1     | 100.0           | SELECT * FROM users;
0.1ms    | 1     | -               | SELECT * FROM databases;
0.0ms    | 1     | -               | SELECT * FROM invoices;
0.0ms    | 1     | -               | SELECT * FROM query_snapshots;
```

pganalyze.com



QUERY	ROLE	AVG TIME (MS)	CALLS / MIN	CACHE HIT %	% OF ALL RUNTIME ▾
`WITH upsert AS (...), all_ids AS (S...`	pgaweb_workers	2.82ms	5574.21	98%	15.96%
`UPDATE "backends" SET seen_at_range...`	pgaweb_workers	102.63ms	118.55	100%	12.37%
`WITH slow_queries AS (...) SELECT ...`	pgaweb_workers	652.29ms	11.93	68%	7.91%
`INSERT INTO "backend_states" (serve...`	pgaweb_workers	0.75ms	9055.81	95%	6.90%
`WITH upsert AS (...), all_ids AS (S...`	pgaweb_workers	259.26ms	21.41	30%	5.64%
`SELECT ... FROM "snapshots" JOIN "s...`	pgaweb_workers	1164.10ms	3.28	70%	3.89%
`UPDATE "queries" q SET last_occurre...`	pgaweb_workers	199.31ms	17.20	93%	3.49%
`UPDATE "queries" q SET last_occurre...`	pgaweb_workers	214.59ms	12.18	93%	2.66%
`WITH servers AS (...), s AS (...), ...`	pgaweb_workers	1196201.31ms	0.00	79%	2.57%

Lock Statistics

`pg_locks`

```
      pid: process ID
           (JOIN to pg_stat_activity.pid!)
 locktype: type of object being locked
     mode: locking type (e.g. AccessExclusive)
  granted: Lock Granted vs Being Waited For
```

…

@LukasFittl

Lock Statistics

`pg_locks`

```
SELECT *
  FROM pg_locks
 WHERE NOT granted
```

@LukasFittl

Lock Statistics

pg_locks

```
SELECT locktype,
       mode,
       COUNT(*)
  FROM pg_locks
 WHERE granted
 GROUP BY 1, 2
```

@LukasFittl

Checkpoint Statistics

`pg_stat_bgwriter`

```
checkpoints_timed: # of scheduled checkpoints
  checkpoints_req: # of requested checkpoints
```

1. Time Between Checkpoints
2. % of Timed Checkpoints

autovacuum

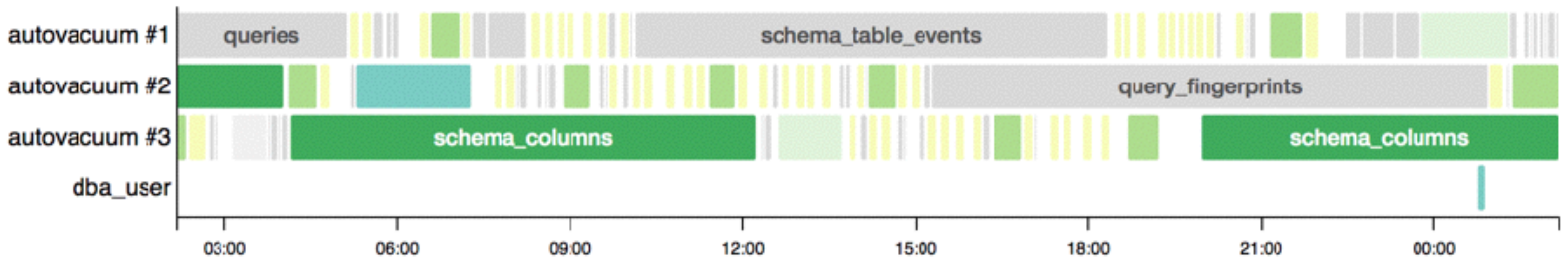
pg_stat_activity

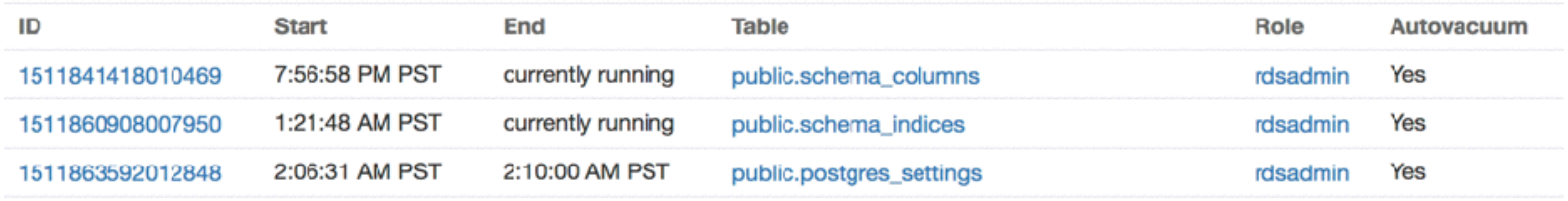
```
=> SELECT pid, query FROM pg_stat_activity
    WHERE query LIKE 'autovacuum: %';

 10469 | autovacuum: VACUUM ANALYZE public.schema_columns
 12848 | autovacuum: VACUUM public.replication_follower_stats
 28626 | autovacuum: VACUUM public.schema_index_stats
       | (to prevent wraparound)
(3 rows)
```

autovacuum

`pg_stat_activity`





ID	Start	End	Table	Role	Autovacuum
1511841418010469	7:56:58 PM PST	currently running	public.schema_columns	rdsadmin	Yes
1511860908007950	1:21:48 AM PST	currently running	public.schema_indices	rdsadmin	Yes
1511863592012848	2:06:31 AM PST	2:10:00 AM PST	public.postgres_settings	rdsadmin	Yes

@LukasFittl

autovacuum

pg_stat_progress_vacuum

relid:	OID of the table
phase:	current VACUUM phase
heap_blks_total:	Heap Blocks Total
heap_blks_scanned:	Heap Blocks Scanned
heap_blks_vacuumed:	Heap Blocks Vacuumed
	…

@LukasFittl

autovacuum

`pg_stat_progress_vacuum`

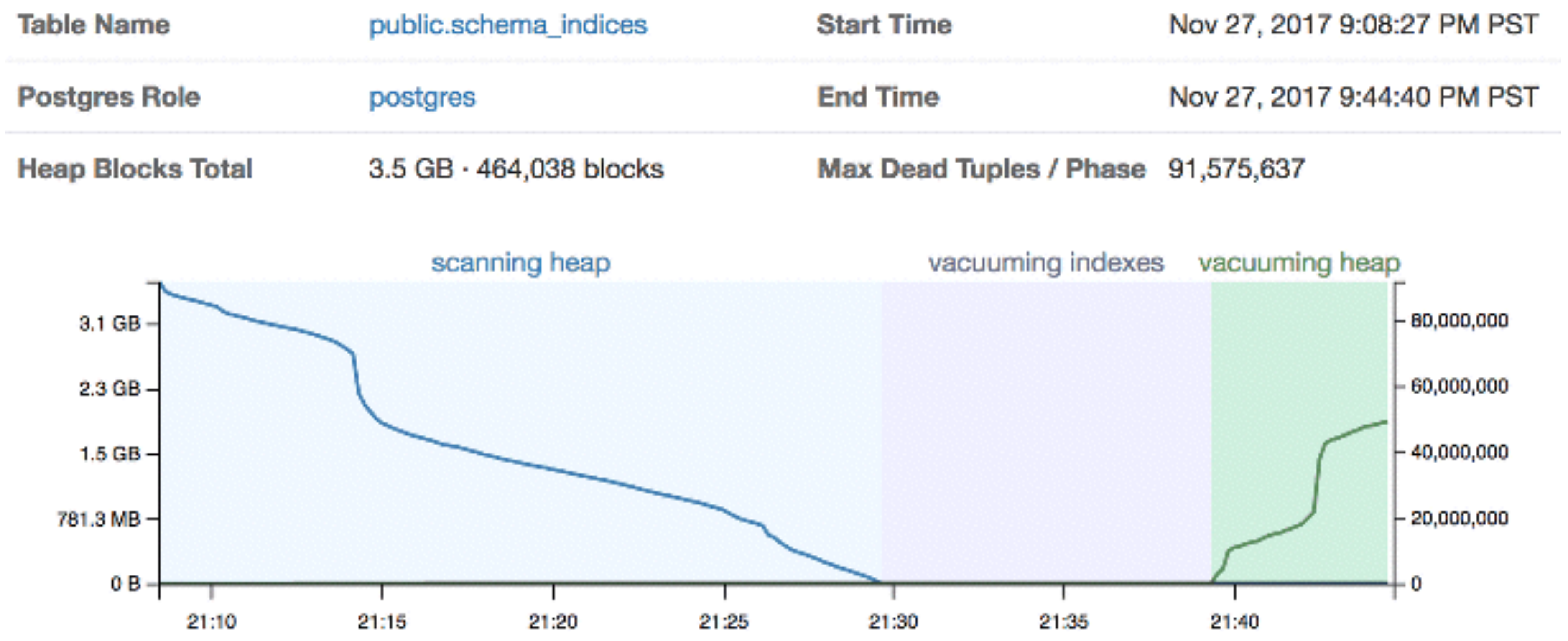


Table Name	public.schema_indices	**Start Time**	Nov 27, 2017 9:08:27 PM PST
Postgres Role	postgres	**End Time**	Nov 27, 2017 9:44:40 PM PST
Heap Blocks Total	3.5 GB · 464,038 blocks	**Max Dead Tuples / Phase**	91,575,637

@LukasFittl

pg_stat_replication

```
    client_addr: ip address of the follower
  backend_start: replication start time
          state: replication state
                 (ideally = streaming)
replay_location: WAL location
```

@LukasFittl

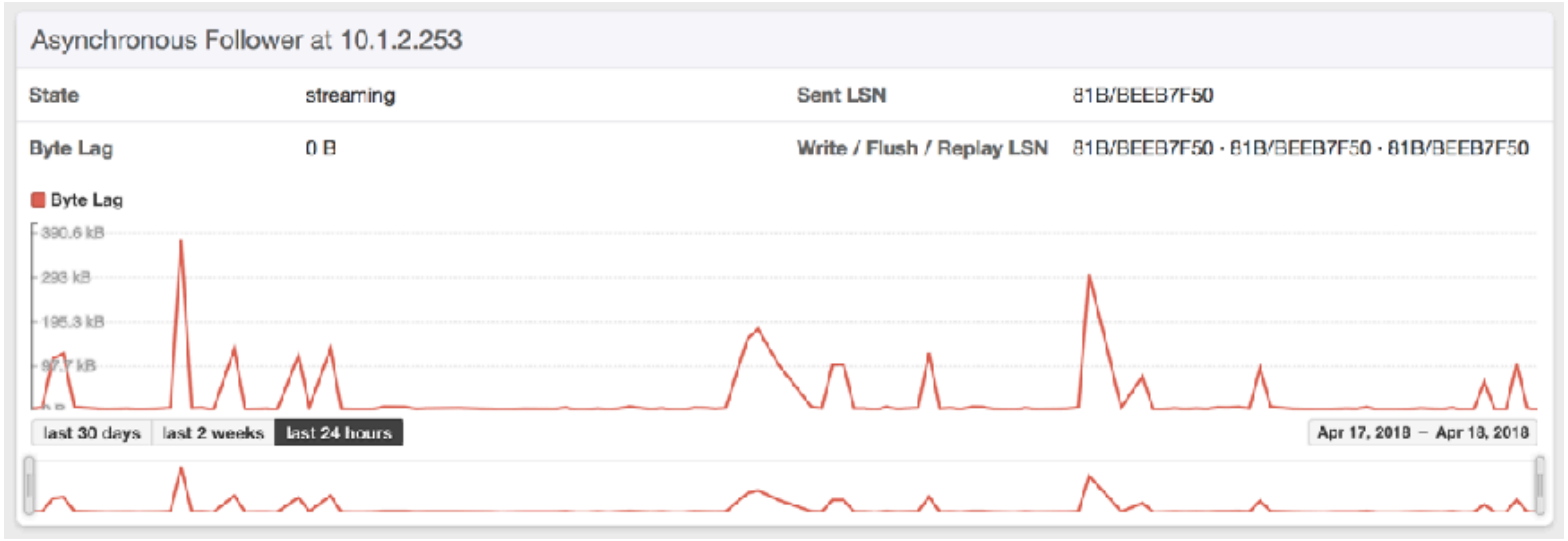
pg_stat_replication

Replication Lag in Bytes, Per Follower

```
SELECT client_addr,
       pg_wal_lsn_diff(
         pg_current_wal_lsn(),
         replay_location)
  FROM pg_stat_replication
```

pg_stat_replication

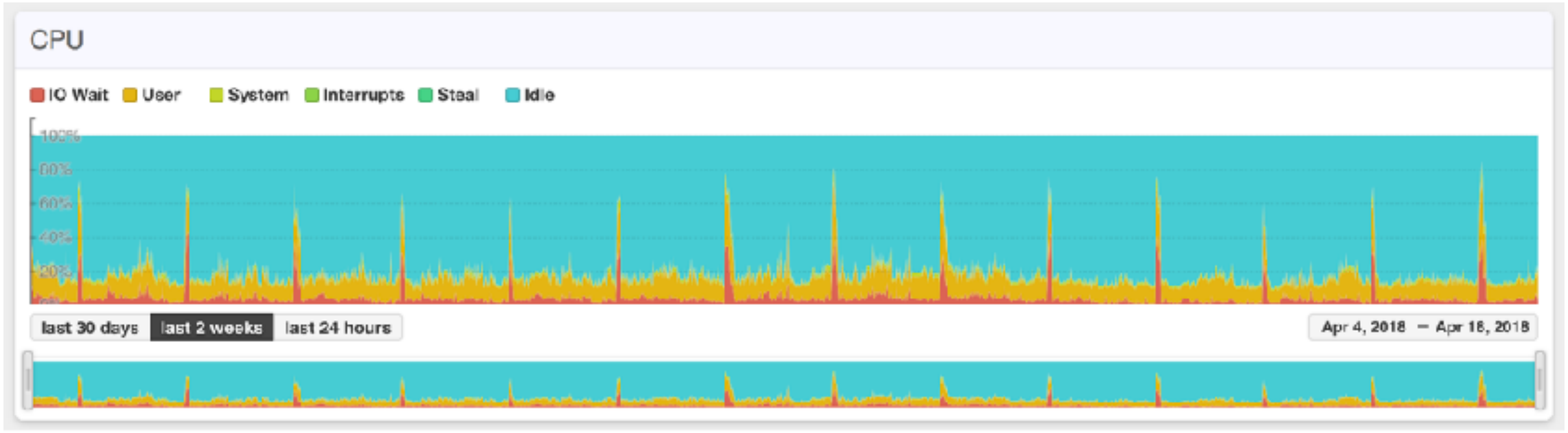
Replication Lag in Bytes, Per Follower

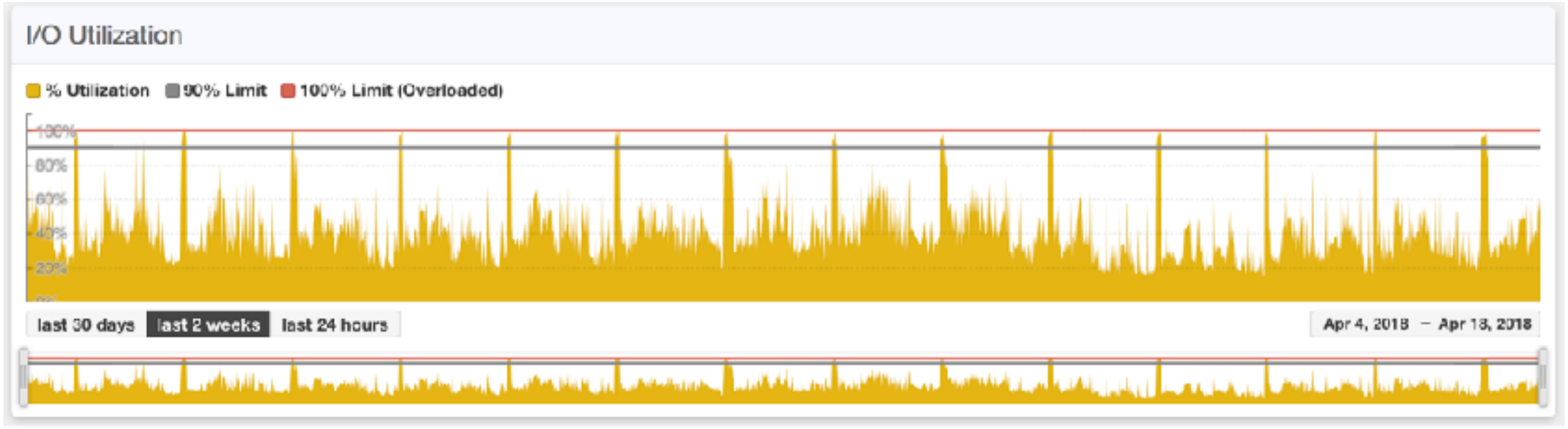

Asynchronous Follower at 10.1.2.253

State	streaming	Sent LSN	81B/BEEB7F50
Byte Lag	0 B	Write / Flush / Replay LSN	81B/BEEB7F50 · 81B/BEEB7F50 · 81B/BEEB7F50

@LukasFittl

CPU & I/O Utilization





@LukasFittl

Statistics That Matter

Two Tables To Remember

Breaking Down High-Level Statistics

Log Events Worth Knowing

Fingerprinting & Tracing Queries

“We had an outage yesterday at 10am - **what happened?**”

Keeping Historic Statistics Data **Is Essential**

DIY Monitoring Hack:

Save **pg_stat_activity** and

pg_stat_database

every 10 seconds

into a separate monitoring database

pg_stat_activity

- Number & State of Connections
- Oldest Query Still Running
- Oldest Transaction Still Open
- Blocked Queries

pg_stat_database

- Transactions Per Second
- Data Read Per Second
- Rows Updated/etc Per Second
- Deadlocks Per Second
- …

@LukasFittl

Statistics That Matter

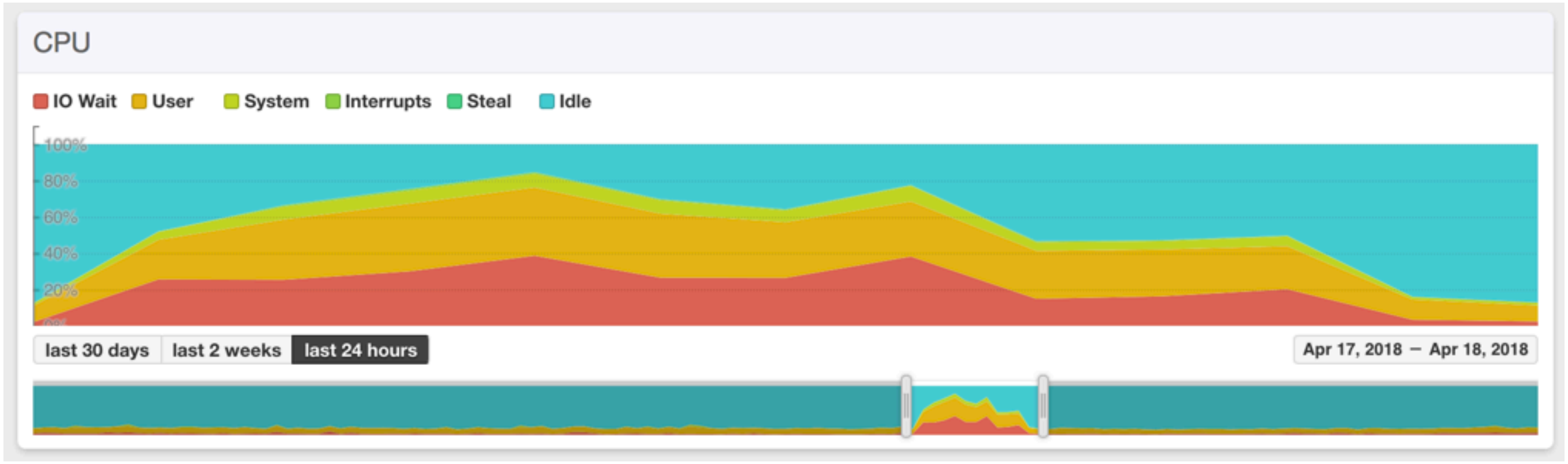
Two Tables To Remember

Breaking Down High-Level Statistics

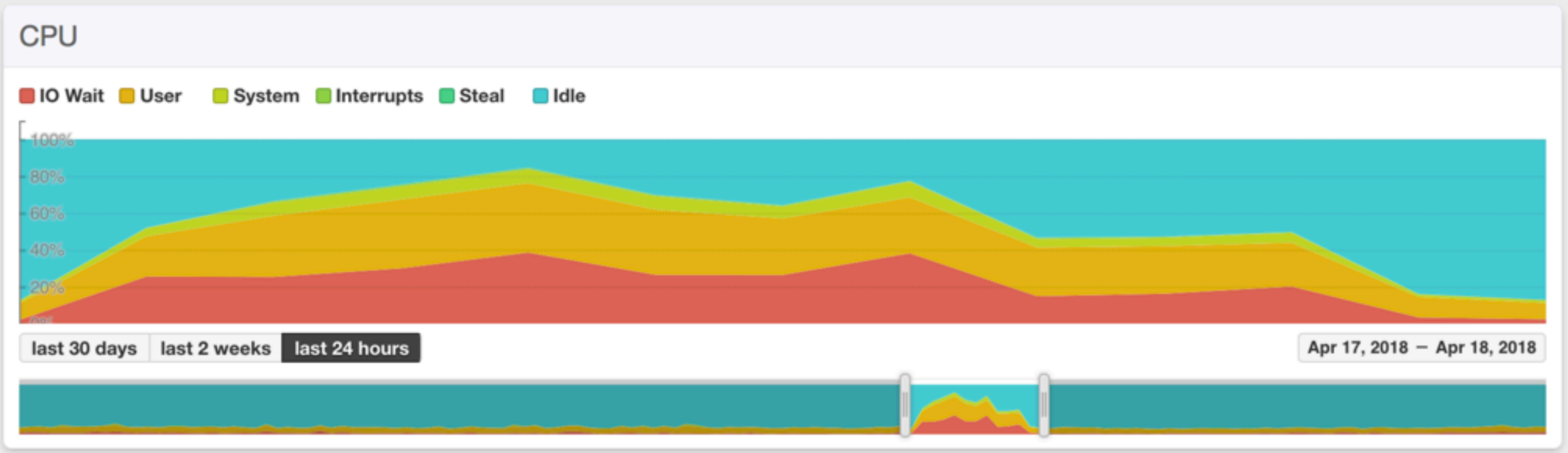
Log Events Worth Knowing

Fingerprinting & Tracing Queries

Ability to Drill Down From "**High CPU Utilization**" To Specific Set of Queries

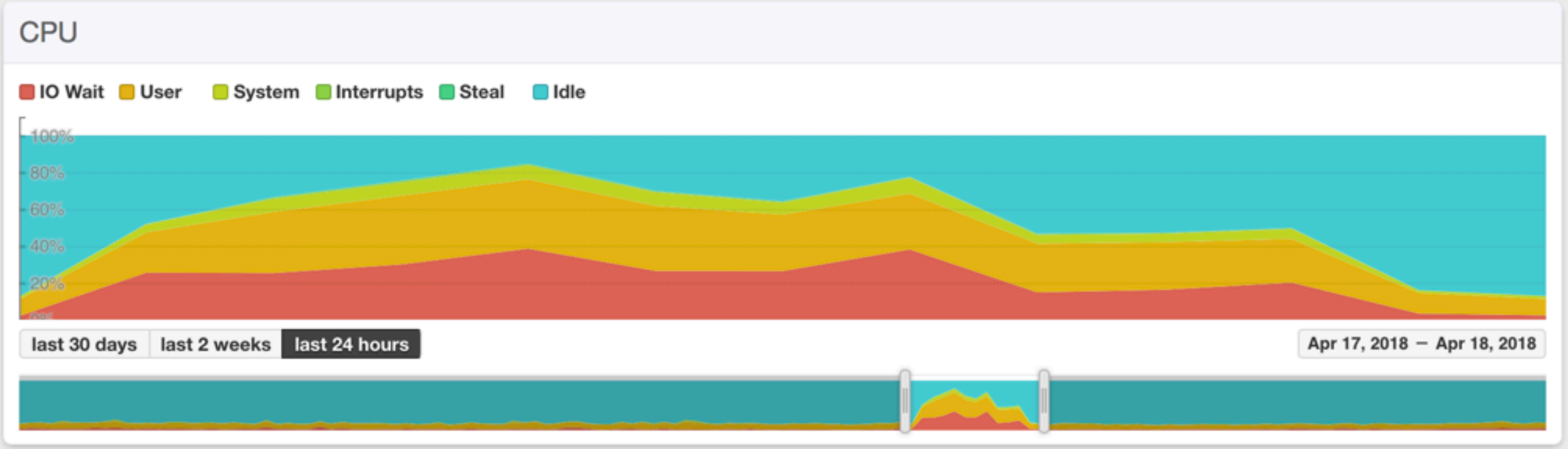

CPU
IO Wait
User
System
Interrupts
Steal
Idle
100%
80%
60%
40%
20%
0%
last 30 days
last 2 weeks
last 24 hours
Apr 17, 2018 – Apr 18, 2018

@LukasFittl



QUERY	ROLE	AVG TIME (MS)	CALLS / MIN	CACHE HIT %	% OF ALL RUNTIME
WITH upsert AS (...), all_ids AS (SELECT …	pgaweb_workers	11.21ms	5853.12	97%	20.09%
WITH servers AS (...), s AS (...), l AS (…	pgaweb_workers	4424695.01ms	0.01	86%	14.35%
INSERT INTO "backend_states" (server_id, …	pgaweb_workers	5.24ms	8777.55	94%	14.09%
UPDATE "backends" SET seen_at_range = tst…	pgaweb_workers	167.02ms	135.74	100%	6.94%
WITH upsert AS (...), all_ids AS (SELECT …	pgaweb_workers	741.30ms	24.99	41%	5.67%
WITH slow_queries AS (...) SELECT ... FRO…	pgaweb_workers	1109.38ms	16.24	30%	5.52%
WITH upsert AS (...), all_ids AS (SELECT …	pgaweb_workers	2.78ms	4175.83	96%	3.56%
SELECT ... FROM "schema_indices" JOIN "sc…	pgaweb_workers	507.85ms	16.59	96%	2.58%
SELECT ... FROM "snapshots" JOIN "system_…	pgaweb_workers	1667.52ms	4.77	84%	2.43%
UPDATE "queries" q SET last_occurred_at =…	pgaweb_workers	363.37ms	19.45	91%	2.16%
UPDATE "queries" q SET last_occurred_at =…	pgaweb_workers	330.39ms	21.38	92%	2.16%
INSERT INTO "backends" (...) VALUES (?)	pgaweb_workers	10.81ms	316.58	88%	1.05%

@LukasFittl



QUERY	ROLE	AVG TIME (MS)	CALLS / MIN	CACHE HIT %	% OF ALL RUNTIME ▾
WITH upsert AS (...), all_ids AS (SELECT …	pgaweb_workers	11.21ms	5853.12	97%	20.09%
WITH servers AS (...), s AS (...), l AS (…	pgaweb_workers	4424695.01ms	0.01	86%	14.35%
INSERT INTO "backend_states" (server_id, …	pgaweb_workers	5.24ms	8777.55	94%	14.09%
UPDATE "backends" SET seen_at_range = tst…	pgaweb_workers	167.02ms	135.74	100%	6.94%
WITH upsert AS (...), all_ids AS (SELECT …	pgaweb_workers	741.30ms	24.99	41%	5.67%
WITH slow_queries AS (...) SELECT ... FRO…	pgaweb_workers	1109.38ms	16.24	30%	5.52%
WITH upsert AS (...), all_ids AS (SELECT …	pgaweb_workers	2.78ms	4175.83	96%	3.56%
SELECT ... FROM "schema_indices" JOIN "sc…	pgaweb_workers	507.85ms	16.59	96%	2.58%
SELECT ... FROM "snapshots" JOIN "system_…	pgaweb_workers	1667.52ms	4.77	84%	2.43%
UPDATE "queries" q SET last_occurred_at =…	pgaweb_workers	363.37ms	19.45	91%	2.16%
UPDATE "queries" q SET last_occurred_at =…	pgaweb_workers	330.39ms	21.38	92%	2.16%
INSERT INTO "backends" (...) VALUES (?)	pgaweb_workers	10.81ms	316.58	88%	1.05%

@LukasFittl

CPU Utilization

↓

pg_stat_statements.**total_runtime**

@LukasFittl

I/O Utilization

↓

pg_stat_statements.**blk_read_time**
pg_stat_statements.**blk_write_time**

@LukasFittl

Cache Hit Ratio %

pg_stat_database.**blks_hit**
pg_stat_database.**blks_read**

↓

pg_stat_statements.**shared_blks_hit**
pg_stat_statements.**shared_blks_read**

@LukasFittl

Temporary Files Written

pg_stat_database.**temp_bytes**

$\downarrow$

pg_stat_statements.**temp_blks_written**

@LukasFittl

Statistics That Matter

Two Tables To Remember

Breaking Down High-Level Statistics

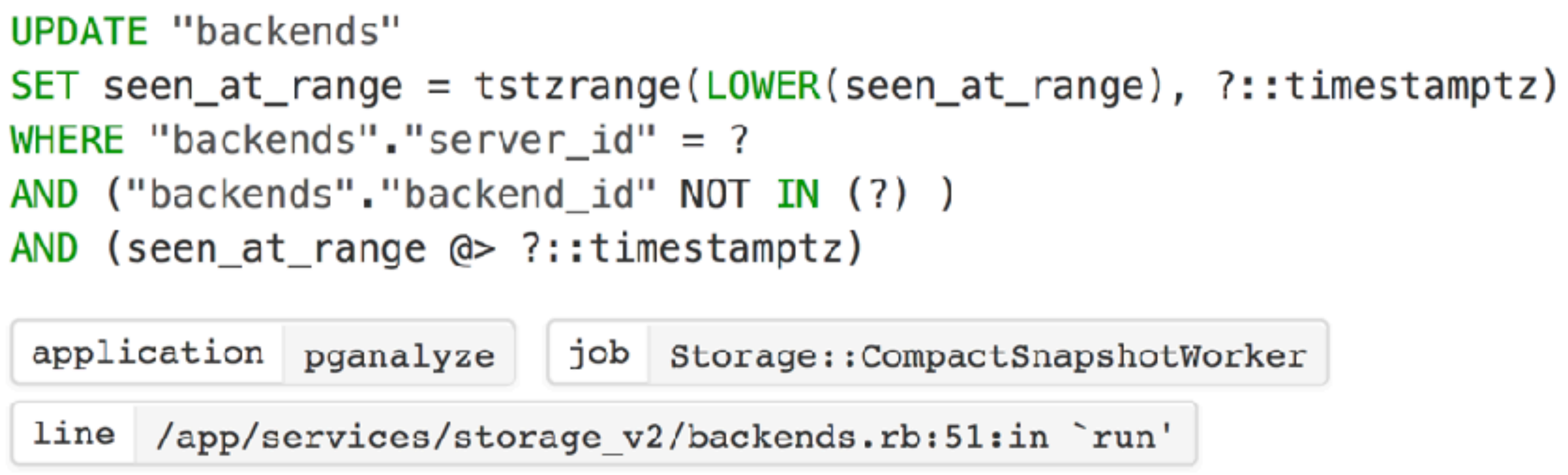
Log Events Worth Knowing

Fingerprinting & Tracing Queries

Slow Queries

log_min_duration_statement = 1000 ms

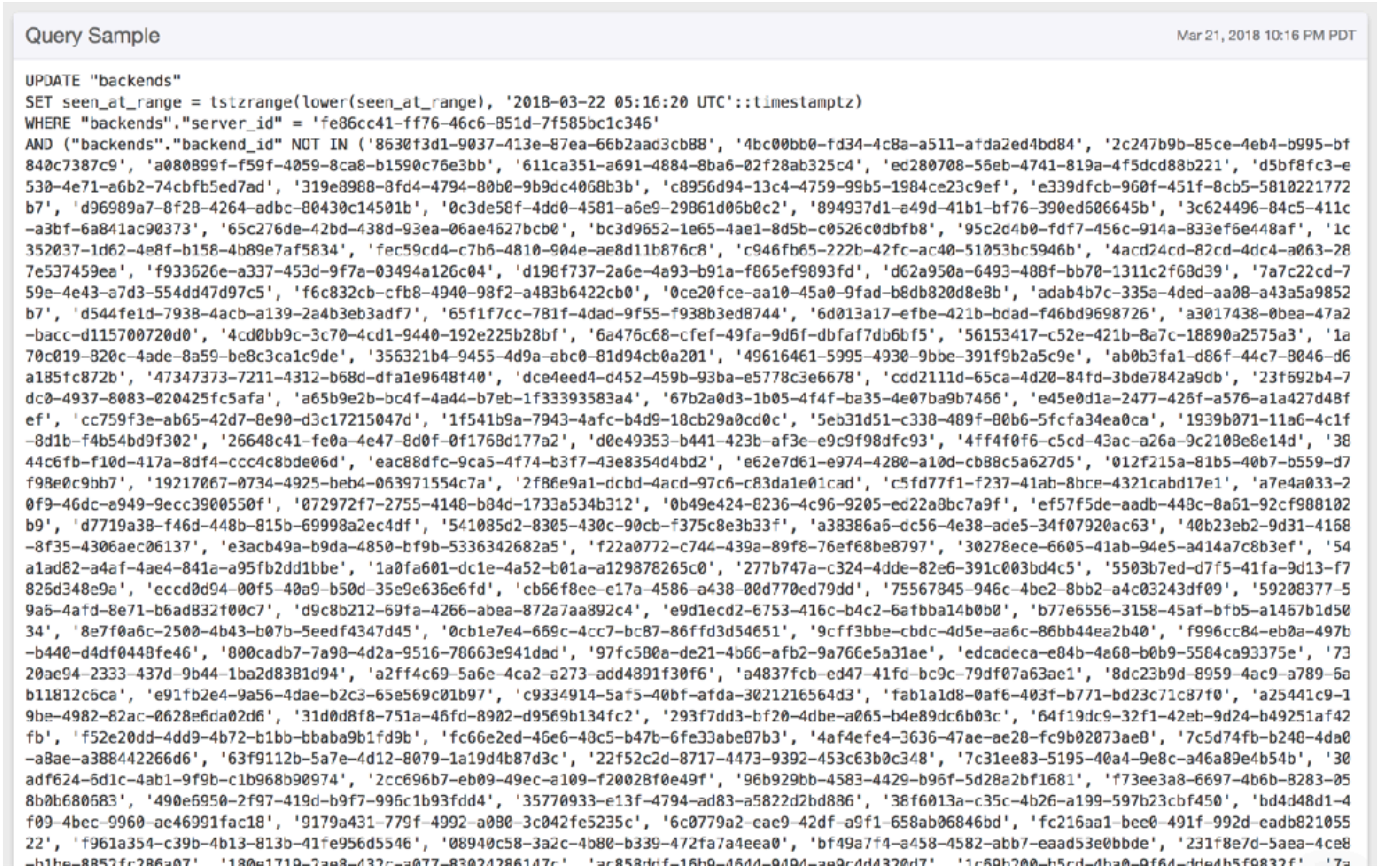
```
LOG: duration: 4079.697 ms execute <unnamed>:
SELECT * FROM x WHERE y = $1 LIMIT $2
DETAIL: parameters: $1 = 'long string', $2 = '1'
```



```sql
UPDATE "backends"
SET seen_at_range = tstzrange(LOWER(seen_at_range), ?::timestamptz)
WHERE "backends"."server_id" = ?
AND ("backends"."backend_id" NOT IN (?) )
AND (seen_at_range @> ?::timestamptz)
```

application	pganalyze	job	Storage::CompactSnapshotWorker
line	/app/services/storage_v2/backends.rb:51:in `run'		

@LukasFittl



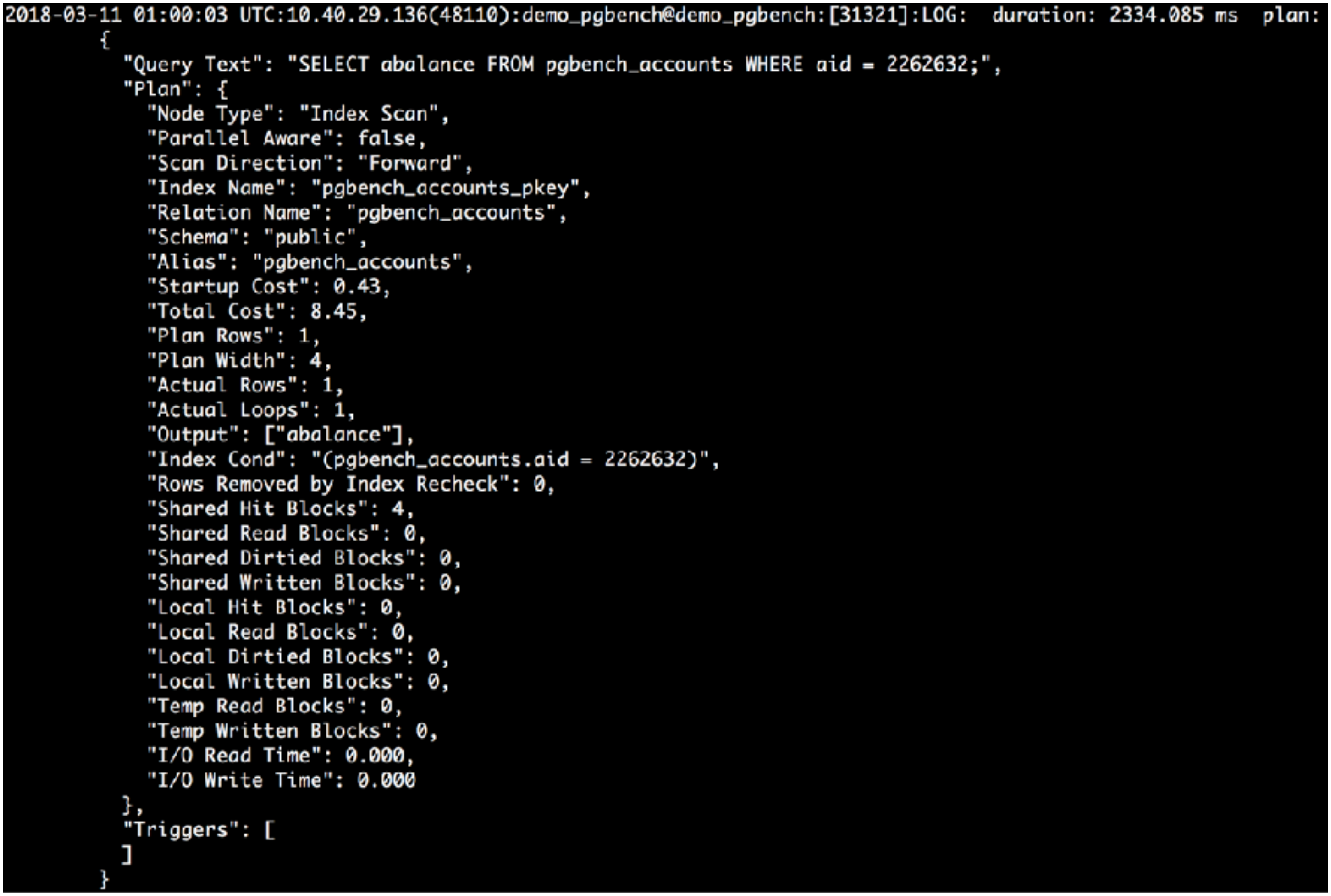
Query Sample

Mar 21, 2018 10:16 PM PDT

```
UPDATE "backends"
SET seen_at_range = tstzrange(lower(seen_at_range), '2018-03-22 05:16:20 UTC'::timestamptz)
WHERE "backends"."server_id" = 'fe86cc41-ff76-46c6-851d-7f585bc1c346'
AND ("backends"."backend_id" NOT IN ('8630f3d1-9037-413e-87ea-66b2aad3cb88', '4bc00bb0-fd34-4c8a-a511-afda2ed4bd84', '2c247b9b-85ce-4eb4-b995-bf
840c7387c9', 'a080899f-f59f-4059-8ca8-b1590c76e3bb', '611ca351-a691-4884-8ba6-02f28ab325c4', 'ed280708-56eb-4741-819a-4f5dcd88b221', 'd5bf8fc3-e
530-4e71-a6b2-74cbfb5ed7ad', '319e8988-8fd4-4794-80b0-9b9dc4068b3b', 'c8956d94-13c4-4759-99b5-1984ce23c9ef', 'e339dfcb-960f-451f-8cb5-5810221772
b7', 'd96989a7-8f28-4264-adbc-80430c14501b', '0c3de58f-4dd0-4581-a6e9-29861d06b0c2', '894937d1-a49d-41b1-bf76-390ed606645b', '3c624496-84c5-411c
-a3bf-6a841ac90373', '65c276de-42bd-438d-93ea-06ae4627bcb0', 'bc3d9652-1e65-4ae1-8d5b-c0526c0dbfb8', '95c2d4b0-fdf7-456c-914a-833ef6e448af', '1c
352037-1d62-4e8f-b158-4b89e7af5834', 'fec59cd4-c7b6-4810-904e-ae8d11b876c8', 'c946fb65-222b-42fc-ac40-51053bc5946b', '4acd24cd-82cd-4dc4-a063-28
7e537459ea', 'f933626e-a337-453d-9f7a-03494a126c04', 'd198f737-2a6e-4a93-b91a-f865ef9893fd', 'd62a950a-6493-488f-bb70-1311c2f68d39', '7a7c22cd-7
59e-4e43-a7d3-554dd47d97c5', 'f6c832cb-cfb8-4940-98f2-a483b6422cb0', '0ce20fce-aa10-45a0-9fad-b8db820d8e8b', 'adab4b7c-335a-4ded-aa08-a43a5a9852
b7', 'd544fe1d-7938-4acb-a139-2a4b3eb3adf7', '65f1f7cc-781f-4dad-9f55-f938b3ed8744', '6d013a17-efbe-421b-bdad-f46bd9698726', 'a3017438-0bea-47a2
-bacc-d115700720d0', '4cd0bb9c-3c70-4cd1-9440-192e225b28bf', '6a476c68-cfef-49fa-9d6f-dbfaf7db6bf5', '56153417-c52e-421b-8a7c-18890a2575a3', '1a
70c019-320c-4ade-8a59-be8c3ca1c9de', '356321b4-9455-4d9a-abc0-81d94cb0a201', '49616461-5995-4930-9bbe-391f9b2a5c9e', 'ab0b3fa1-d86f-44c7-8046-d6
a185fc872b', '47347373-7211-4312-b68d-dfa1e9648f40', 'dce4eed4-d452-459b-93ba-e5778c3e6678', 'cdd2111d-65ca-4d20-84fd-3bde7842a9db', '23f692b4-7
dc0-4937-8083-020425fc5afa', 'a65b9e2b-bc4f-4a44-b7eb-1f33393583a4', '67b2a0d3-1b05-4f4f-ba35-4e07ba9b7466', 'e45e0d1a-2477-426f-a576-a1a427d48f
ef', 'cc759f3e-ab65-42d7-8e90-d3c17215047d', '1f541b9a-7943-4afc-b4d9-18cb29a0cd0c', '5eb31d51-c338-489f-80b6-5fcfa34ea0ca', '1939b071-11a6-4c1f
-8d1b-f4b54bd9f302', '26648c41-fe0a-4e47-8d0f-0f1768d177a2', 'd0e49353-b441-423b-af3e-e9c9f98dfc93', '4ff4f0f6-c5cd-43ac-a26a-9c2108e8e14d', '38
44c6fb-f10d-417a-8df4-ccc4c8bde06d', 'eac88dfc-9ca5-4f74-b3f7-43e8354d4bd2', 'e62e7d61-e974-4280-a10d-cb88c5a627d5', '012f215a-81b5-40b7-b559-d7
f98e0c9bb7', '19217067-0734-4925-beb4-063971554c7a', '2f86e9a1-dcbd-4acd-97c6-c83da1e01cad', 'c5fd77f1-f237-41ab-8bce-4321cabd17e1', 'a7e4a033-2
0f9-46dc-a949-9ecc3900550f', '072972f7-2755-4148-b84d-1733a534b312', '0b49e424-8236-4c96-9205-ed22a8bc7a9f', 'ef57f5de-aadb-448c-8a61-92cf988102
b9', 'd7719a38-f46d-448b-815b-69998a2ec4df', '541085d2-8305-430c-90cb-f375c8e3b33f', 'a38386a6-dc56-4e38-ade5-34f07920ac63', '40b23eb2-9d31-4168
-8f35-4306aec06137', 'e3acb49a-b9da-4850-bf9b-5336342682a5', 'f22a0772-c744-439a-89f8-76ef68be8797', '30278ece-6605-41ab-94e5-a414a7c8b3ef', '54
a1ad82-a4af-4ae4-841a-a95fb2dd1bbe', '1a0fa601-dc1e-4a52-b01a-a129878265c0', '277b747a-c324-4dde-82e6-391c003bd4c5', '5503b7ed-d7f5-41fa-9d13-f7
826d348e9a', 'eccd0d94-00f5-40a9-b50d-35e9e636e6fd', 'cb66f8ee-e17a-4586-a438-00d770ed79dd', '75567845-946c-4be2-8bb2-a4c03243df09', '59208377-5
9a6-4afd-8e71-b6ad832f00c7', 'd9c8b212-69fa-4266-abea-872a7aa892c4', 'e9d1ecd2-6753-416c-b4c2-6afbba14b0b0', 'b77e6556-3158-45af-bfb5-a1467b1d50
34', '8e7f0a6c-2500-4b43-b07b-5eedf4347d45', '0cb1e7e4-669c-4cc7-bc87-86ffd3d54651', '9cff3bbe-cbdc-4d5e-aa6c-86bb44ea2b40', 'f996cc84-eb0a-497b
-b440-d4df0448fe46', '800cadb7-7a98-4d2a-9516-78663e941dad', '97fc580a-de21-4b66-afb2-9a766e5a31ae', 'edcadeca-e84b-4a68-b0b9-5584ca93375e', '73
20ae94-2333-437d-9b44-1ba2d8381d94', 'a2ff4c69-5a6e-4ca2-a273-add4891f30f6', 'a4837fcb-ed47-41fd-bc9c-79df07a63ae1', '8dc23b9d-8959-4ac9-a789-6a
b11812c6ca', 'e91fb2e4-9a56-4dae-b2c3-65e569c01b97', 'c9334914-5af5-40bf-afda-3021216564d3', 'fab1a1d8-0af6-403f-b771-bd23c71c87f0', 'a25441c9-1
9be-4982-82ac-0628e6da02d6', '31d0d8f8-751a-46fd-8902-d9569b134fc2', '293f7dd3-bf20-4dbe-a065-b4e89dc6b03c', '64f19dc9-32f1-42eb-9d24-b49251af42
fb', 'f52e20dd-4dd9-4b72-b1bb-bbaba9b1fd9b', 'fc66e2ed-46e6-48c5-b47b-6fe33abe87b3', '4af4efe4-3636-47ae-ae28-fc9b02073ae8', '7c5d74fb-b248-4da0
-a8ae-a388442266d6', '63f9112b-5a7e-4d12-8079-1a19d4b87d3c', '22f52c2d-8717-4473-9392-453c63b0c348', '7c31ee83-5195-40a4-9e8c-a46a89e4b54b', '30
adf624-6d1c-4ab1-9f9b-c1b968b90974', '2cc696b7-eb09-49ec-a109-f20028f0e49f', '96b929bb-4583-4429-b96f-5d28a2bf1681', 'f73ee3a8-6697-4b6b-8283-05
8b0b680683', '490e6950-2f97-419d-b9f7-996c1b93fdd4', '35770933-e13f-4794-ad83-a5822d2bd886', '38f6013a-c35c-4b26-a199-597b23cbf450', 'bd4d48d1-4
f09-4bec-9960-ae46991fac18', '9179a431-779f-4992-a080-3c042fe5235c', '6c0779a2-eae9-42df-a9f1-658ab06846bd', 'fc216aa1-bee0-491f-992d-eadb821055
22', 'f961a354-c39b-4b13-813b-41fe956d5546', '08940c58-3a2c-4b80-b339-472fa7a4eea0', 'bf49a7f4-a458-4582-abb7-eaad53e0bbde', '231f8e7d-5aea-4ce8
-b1be-8852fc286a07', '180e1719-2ae8-432c-a077-83024286147c', 'ac858ddf-16b9-4644-9494-ae9c4d4320d7', '1c69b200-b5cd-4ba0-9f64-dde4b5f9832f', '7a
```

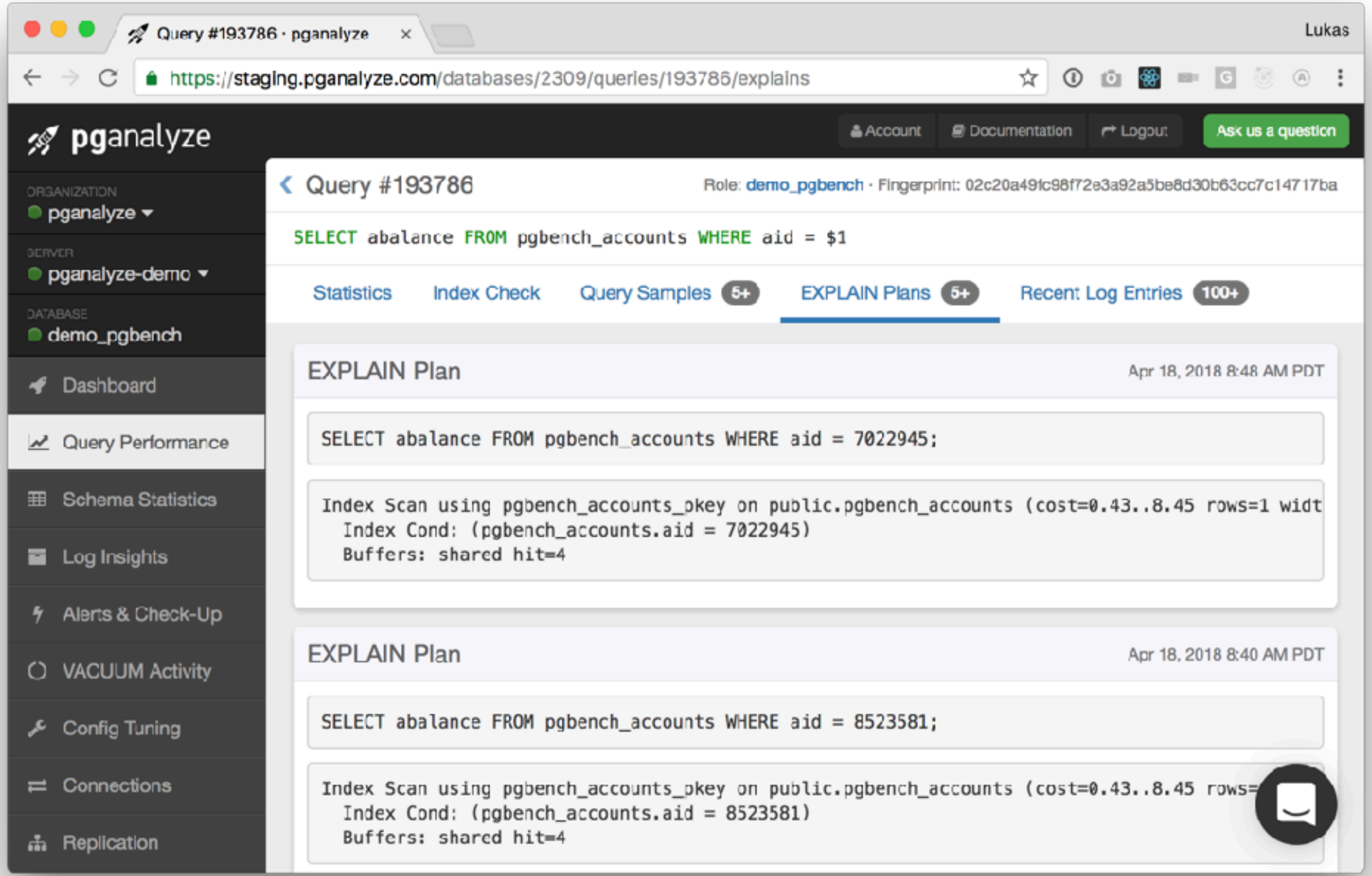
@LukasFittl

auto_explain
logs the query plan
for specific slow queries



```
2018-03-11 01:00:03 UTC:10.40.29.136(48110):demo_pgbench@demo_pgbench:[31321]:LOG:  duration: 2334.085 ms  plan:
        {
          "Query Text": "SELECT abalance FROM pgbench_accounts WHERE aid = 2262632;",
          "Plan": {
            "Node Type": "Index Scan",
            "Parallel Aware": false,
            "Scan Direction": "Forward",
            "Index Name": "pgbench_accounts_pkey",
            "Relation Name": "pgbench_accounts",
            "Schema": "public",
            "Alias": "pgbench_accounts",
            "Startup Cost": 0.43,
            "Total Cost": 8.45,
            "Plan Rows": 1,
            "Plan Width": 4,
            "Actual Rows": 1,
            "Actual Loops": 1,
            "Output": ["abalance"],
            "Index Cond": "(pgbench_accounts.aid = 2262632)",
            "Rows Removed by Index Recheck": 0,
            "Shared Hit Blocks": 4,
            "Shared Read Blocks": 0,
            "Shared Dirtied Blocks": 0,
            "Shared Written Blocks": 0,
            "Local Hit Blocks": 0,
            "Local Read Blocks": 0,
            "Local Dirtied Blocks": 0,
            "Local Written Blocks": 0,
            "Temp Read Blocks": 0,
            "Temp Written Blocks": 0,
            "I/O Read Time": 0.000,
            "I/O Write Time": 0.000
          },
          "Triggers": [
          ]
        }
```

@LukasFittl



Query #193786

Role: demo_pgbench · Fingerprint: 02c20a49fc98f72e3a92a5be8d30b63cc7c14717ba

```
SELECT abalance FROM pgbench_accounts WHERE aid = $1
```

EXPLAIN Plan — Apr 18, 2018 8:48 AM PDT

```
SELECT abalance FROM pgbench_accounts WHERE aid = 7022945;
```

```
Index Scan using pgbench_accounts_pkey on public.pgbench_accounts (cost=0.43..8.45 rows=1 widt
  Index Cond: (pgbench_accounts.aid = 7022945)
  Buffers: shared hit=4
```

EXPLAIN Plan — Apr 18, 2018 8:40 AM PDT

```
SELECT abalance FROM pgbench_accounts WHERE aid = 8523581;
```

```
Index Scan using pgbench_accounts_pkey on public.pgbench_accounts (cost=0.43..8.45 rows=
  Index Cond: (pgbench_accounts.aid = 8523581)
  Buffers: shared hit=4
```

@LukasFittl

Cancelled Queries

```
    ERROR: canceling statement due to
           statement timeout
STATEMENT: SELECT 1

    ERROR: canceling statement due to
           user request
STATEMENT: SELECT 1

           …
```

@LukasFittl

Lock Waits

```
log_lock_waits = on
```

```
LOG: process 20679 still waiting for ExclusiveLock on tuple (566,1) of relation 16421 after 1000.115 ms
LOG: process 20678 still waiting for ExclusiveLock on tuple (566,1) of relation 16421 after 1000.126 ms
LOG: process 15533 still waiting for ExclusiveLock on tuple (566,1) of relation 16421 1000.129 ms
LOG: process 20663 still waiting for ExclusiveLock on tuple (566,1) of relation 16421 1000.100 ms
LOG: process 15537 still waiting for ExclusiveLock on tuple (566,1) of relation 16421 1000.130 ms
LOG: process 15536 still waiting for ExclusiveLock on tuple (566,1) of relation 16421 1000.222 ms
LOG: process 20734 still waiting for ExclusiveLock on tuple (566,1) of relation 16421 1000.130 ms
LOG: process 15538 still waiting for ExclusiveLock on tuple (566,1) of relation 16421 1000.136 ms
LOG: process 15758 still waiting for ShareLock on transaction 250175899 after 1000.073 ms
```

@LukasFittl

archive_command Failures

```
LOG: archive command failed with
exit code 1
DETAIL: The failed archive command
was: /my_backup_script.sh pg_xlog/
00000001000025DFA00000023
```

@LukasFittl

Out of Memory

```
 ERROR: out of memory
DETAIL: Failed on request of size 408028.
 QUERY: SELECT 1 ...
```

@LukasFittl

Out of Connections

```
FATAL: remaining connection slots
are reserved for non-replication
superuser connections
```

Server Crash / Segfault

```
LOG: server process (PID 660) was
terminated by signal 6: Aborted
DETAIL: Failed process was running:
SELECT pg_advisory_lock(1, 2);
LOG: terminating any other active
server processes
WARNING: terminating connection
because of crash of another server
process
…
```

TXID Wraparound

```
WARNING: database “mydb" must
         be vacuumed within 938860
         transactions

   HINT: To avoid a database
         shutdown, execute a full-
         database VACUUM in
         “mydb".
```

@LukasFittl

TXID Wraparound

```
ERROR: database is not accepting
       commands to avoid wraparound
       data loss in database “mydb”

 HINT: Stop the postmaster and use a
       standalone backend to vacuum
       that database. You might also
       need to commit or roll back
       old prepared transactions.
```

Statistics That Matter

Two Tables To Remember

Breaking Down High-Level Statistics

Log Events Worth Knowing

Fingerprinting & Tracing Queries

Fingerprinting
Identifying & Grouping Queries

A

```
SELECT a, b
  FROM public.test
 WHERE col = ‘value’
```

@LukasFittl

A

```
SELECT a, b
  FROM public.test
 WHERE col = ‘value’
```

B

```
SELECT a, b
  FROM public.test
 WHERE col = ‘other_value’
```

@LukasFittl

A

```
SELECT a, b
  FROM public.test
 WHERE col = ?
```

A

```
SELECT a, b
  FROM public.test
 WHERE col = ?
```

A

```
SELECT a, b
  FROM public.test
 WHERE col = ?
```

B

```
SELECT a, b -- COMMENT
  FROM public.test
 WHERE col = ?
```

@LukasFittl

pg_stat_statements

```
# SELECT queryid, query FROM pg_stat_statements;
  queryid   |                    query
------------+---------------------------------------
 1115711211 | SELECT a, b FROM public.test WHERE col = $1
(1 row)
```

@LukasFittl

A

```
SELECT a, b
  FROM public.test
 WHERE col = ?
```

queryid = 1115711211

A

```
SELECT a, b -- COMMENT
  FROM public.test
 WHERE col = ?
```

queryid = 1115711211

@LukasFittl

A

```
SELECT a, b
  FROM public.test
 WHERE col = ?
```

queryid = 1115711211

B

```
SELECT b, a -- COMMENT
  FROM public.test
 WHERE col = ?
```

queryid = 2511327719

@LukasFittl

pg_query

```
irb> PgQuery.fingerprint(
       ‘SELECT a, b FROM public.test WHERE col = $1’)

=> 0254f1e78f2d47b258d7b022f3dfa5794351a75128
```

@LukasFittl

A

```
SELECT a, b
  FROM public.test
 WHERE col = ?
```

fingerprint = 0254f1e78f2d47b258d7b022f3dfa5794351a75128

A

```
SELECT b, a /* COMMENT */
  FROM public.test
 WHERE col = ?
```

fingerprint = 0254f1e78f2d47b258d7b022f3dfa5794351a75128

@LukasFittl

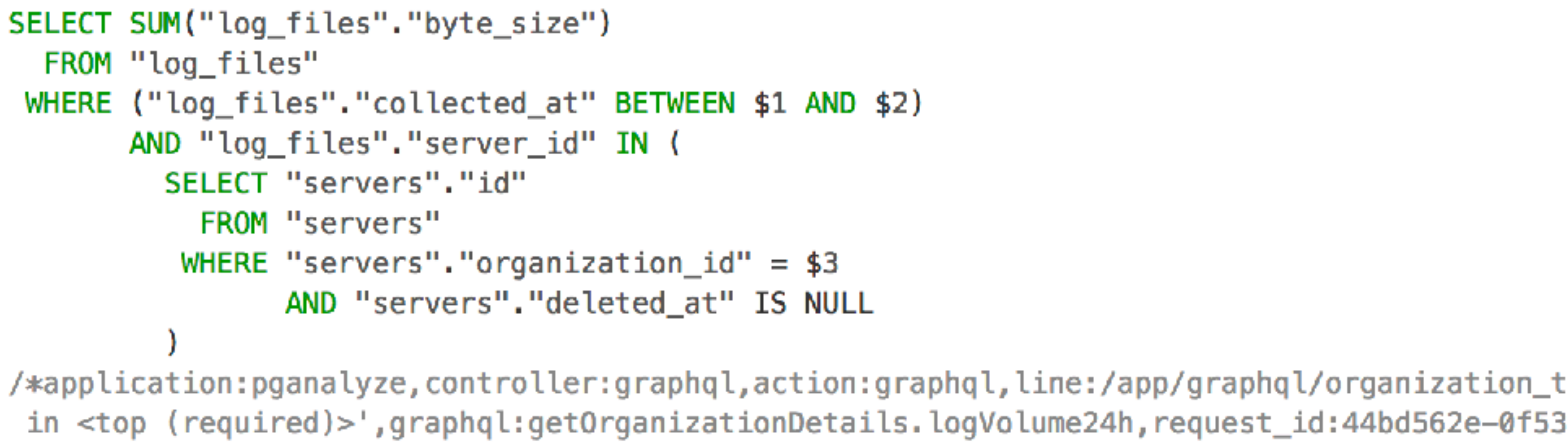
PgQuery.fingerprint

- Based on Postgres Parsetree
- Table names, not OIDs
- Identical across servers & Postgres versions

https://github.com/lfittl/libpg_query/wiki/Fingerprinting

Tracing Queries
Based On Their **Query Origin**

```
SELECT SUM("log_files"."byte_size")
  FROM "log_files"
 WHERE ("log_files"."collected_at" BETWEEN $1 AND $2)
       AND "log_files"."server_id" IN (
         SELECT "servers"."id"
           FROM "servers"
          WHERE "servers"."organization_id" = $3
                AND "servers"."deleted_at" IS NULL
         )
```



```
SELECT SUM("log_files"."byte_size")
  FROM "log_files"
 WHERE ("log_files"."collected_at" BETWEEN $1 AND $2)
       AND "log_files"."server_id" IN (
         SELECT "servers"."id"
           FROM "servers"
          WHERE "servers"."organization_id" = $3
                AND "servers"."deleted_at" IS NULL
         )
/*application:pganalyze,controller:graphql,action:graphql,line:/app/graphql/organization_t
 in <top (required)>',graphql:getOrganizationDetails.logVolume24h,request_id:44bd562e-0f53
```

@LukasFittl

```
application: pganalyze
 controller: graphql
     action: graphql
       line: /app/graphql/organization_type.rb …
    graphql: getOrganizationDetails.logVolume24h
 request_id: 44bd562e-0f53-453f-831f-498e61ab6db5
```

github.com/basecamp/**marginalia**

Automatic
Query Annotations For Ruby on Rails

3 Take-Aways

1. Collect Historic Metrics
2. Focus on Drill-Down To Query Level
3. Annotate Your Queries With Their Origin

Thanks!

Monitor Your Postgres:

pganalyze.com

Scale Your Postgres:

citusdata.com

@LukasFittl