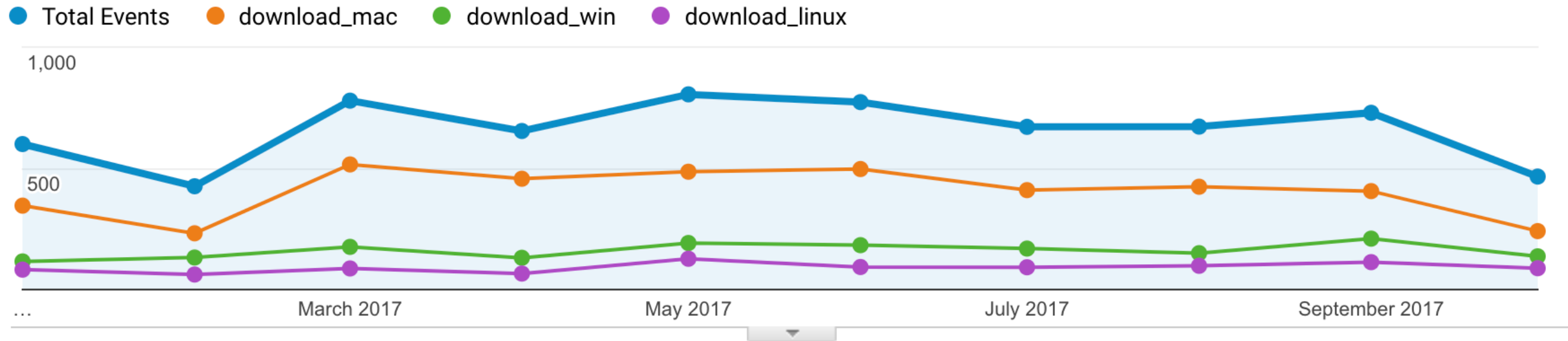


SQL Tabs

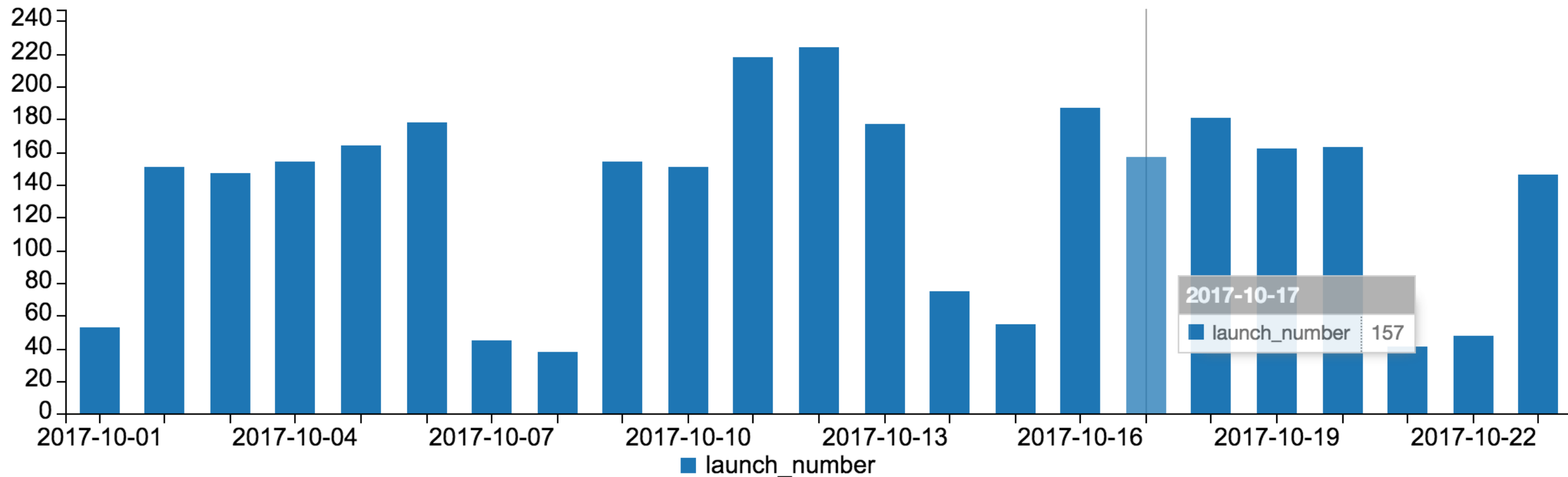
Sasha
Warsaw
2017

- What is SQL Tabs and what it isn't
- Main concepts
- How we can collaborate
- Some thoughts about the future

~700 downloads per month



~160 launches per day



SQL Tabs

[docker dev] [docker dev] [docker dev] +

postgres://postgres:admin@localhost:5432 --- docker dev

```
1
2 DROP TABLE IF EXISTS test;
3
4 CREATE TABLE IF NOT EXISTS test (id SERIAL, val TEXT);
5
6 INSERT INTO test (val) SELECT generate_series(1, 5);
7
8 SELECT * FROM test;
```

Time: 11.9 ms share

notice: table "test" does not exist, skipping

DROP TABLE

CREATE TABLE

INSERT 0 5

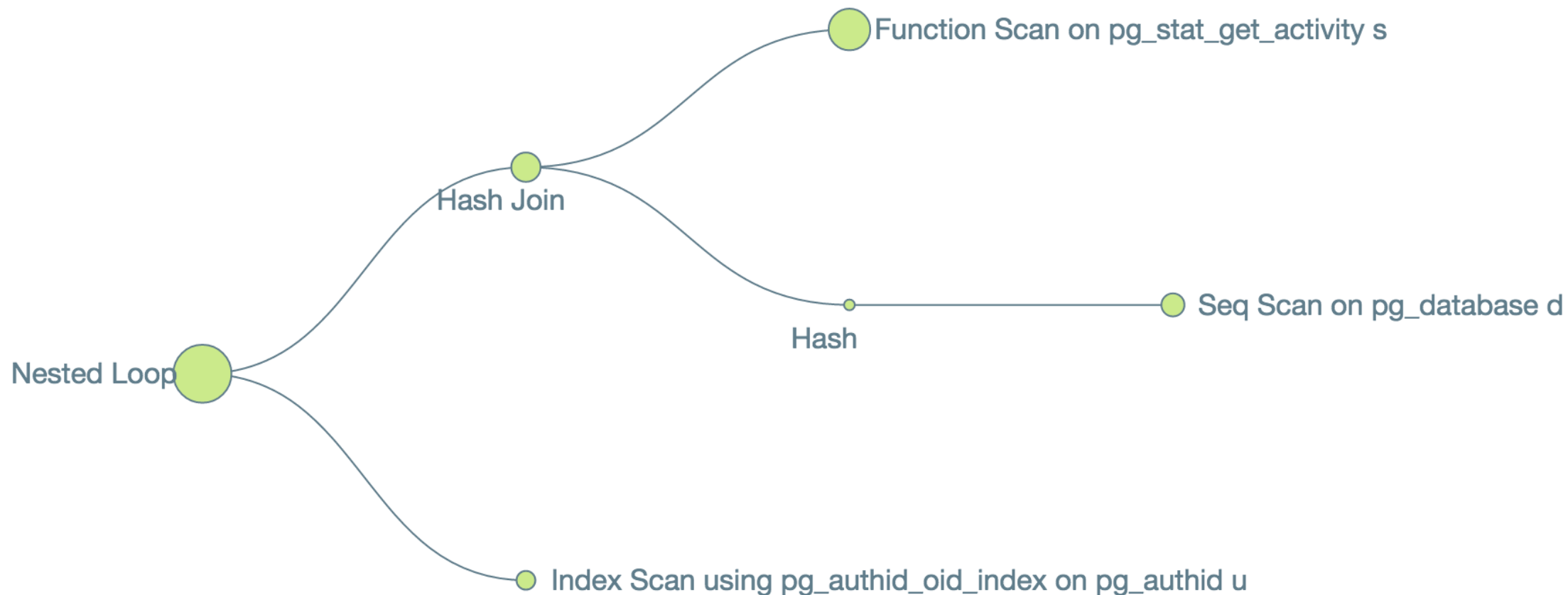
(5 rows)

#	id	val
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5



QUERY PLAN

1	Nested Loop (cost=1.17..2.99 rows=1 width=408) (actual time=0.290..0.600 rows=7 loops=1)
2	➔ Hash Join (cost=1.05..2.44 rows=2 width=344) (actual time=0.268..0.343 rows=7 loops=1)
3	Hash Cond: (s.datid = d.oid)
4	➔ Function Scan on pg_stat_get_activity s (cost=0.00..1.00 rows=100 width=280) (actual time=0.116..0.139 rows=7 loops=1)
5	➔ Hash (cost=1.02..1.02 rows=2 width=68) (actual time=0.106..0.106 rows=3 loops=1)
6	Buckets: 1024 Batches: 1 Memory Usage: 9kB
7	➔ Seq Scan on pg_database d (cost=0.00..1.02 rows=2 width=68) (actual time=0.009..0.077 rows=3 loops=1)
8	➔ Index Scan using pg_authid_oid_index on pg_authid u (cost=0.13..0.27 rows=1 width=68) (actual time=0.006..0.009 rows=1 loops=7)
9	Index Cond: (oid = s.usersysid)
10	Planning time: 0.363 ms
11	Execution time: 0.781 ms



Key Principles

- PSQL is a gold standard
- Engage SQL
- Experts friendly
- Rich Output (SQL Documents)

Let's make components

The screenshot displays the OmniDB application window. The title bar reads "OmniDB". The interface includes a "Connections" panel on the left with a tab for "(ddocker) postgres@postgres localhost:5432" and a "PostgreSQL" icon. The main area shows a "Query" editor with the SQL statement: `1 select * from pg_stat_activity`. Below the query editor, the execution status is shown: "Start time: 10/23/2017 20:30:45 Duration: 8.64 ms". The "Explain" tab is selected, displaying the "QUERY PLAN" for the query. The plan is a nested loop join, with the following steps:

- 1 Nested Loop (cost=1.17..2.99 rows=1 width=408)
- 2 Hash Join (cost=1.05..2.44 rows=2 width=344)
- 3 Hash Cond: (s.datid = d.oid)
- 4 Function Scan on pg_stat_get_activity s (cost=0.00..1.00 rows=100 width=280)
- 5 Hash (cost=1.02..1.02 rows=2 width=68)
- 6 Seq Scan on pg_database d (cost=0.00..1.02 rows=2 width=68)
- 7 Index Scan using pg_authid_oid_index on pg_authid u (cost=0.13..0.27 rows=1 width=68)
- 8 Index Cond: (oid = s.usysid)

The "Explain" tab and the query plan details are highlighted with a red rectangular border.

Let's make components

explain.sqltabs.com

 **SQL Tabs Explain**

```
Nested Loop (cost=1.17..2.99 rows=1 width=408) (actual time=0.127..0.316 rows=8 loops=1)
-> Hash Join (cost=1.05..2.44 rows=2 width=344) (actual time=0.108..0.172 rows=8 loops=1)
    Hash Cond: (s.datid = d.oid)
    -> Function Scan on pg_stat_get_activity s (cost=0.00..1.00 rows=100 width=280) (actual time=0.057..0.079 rows=8 loops=1)
    -> Hash (cost=1.02..1.02 rows=2 width=68) (actual time=0.033..0.033 rows=3 loops=1)
        Buckets: 1024 Batches: 1 Memory Usage: 9kB
        -> Seq Scan on pg_database d (cost=0.00..1.02 rows=2 width=68) (actual time=0.009..0.017 rows=3 loops=1)
    -> Index Scan using pg_authid_oid_index on pg_authid u (cost=0.13..0.27 rows=1 width=68) (actual time=0.004..0.007 rows=1 loops=8)
        Index Cond: (oid = s.usersysid)
Planning time: 0.150 ms
```

#	QUERY PLAN
1	Nested Loop (cost=1.17..2.99 rows=1 width=408) (actual time=0.127..0.316 rows=8 loops=1)
2	➡ Hash Join (cost=1.05..2.44 rows=2 width=344) (actual time=0.108..0.172 rows=8 loops=1)
3	Hash Cond: (s.datid = d.oid)
4	➡ Function Scan on pg_stat_get_activity s (cost=0.00..1.00 rows=100 width=280) (actual time=0.057..0.079 rows=8 loops=1)
5	➡ Hash (cost=1.02..1.02 rows=2 width=68) (actual time=0.033..0.033 rows=3 loops=1)
6	Buckets: 1024 Batches: 1 Memory Usage: 9kB
7	➡ Seq Scan on pg_database d (cost=0.00..1.02 rows=2 width=68) (actual time=0.009..0.017 rows=3 loops=1)
8	➡ Index Scan using pg_authid_oid_index on pg_authid u (cost=0.13..0.27 rows=1 width=68) (actual time=0.004..0.007 rows=1 loops=8)
9	Index Cond: (oid = s.usersysid)
10	Planning time: 0.150 ms
11	Execution time: 0.379 ms

Future

- Data Visualization and SQL Documents
- Secure and manageable reports sharing
- Reusable UI Components
- Fat clients and Javascript