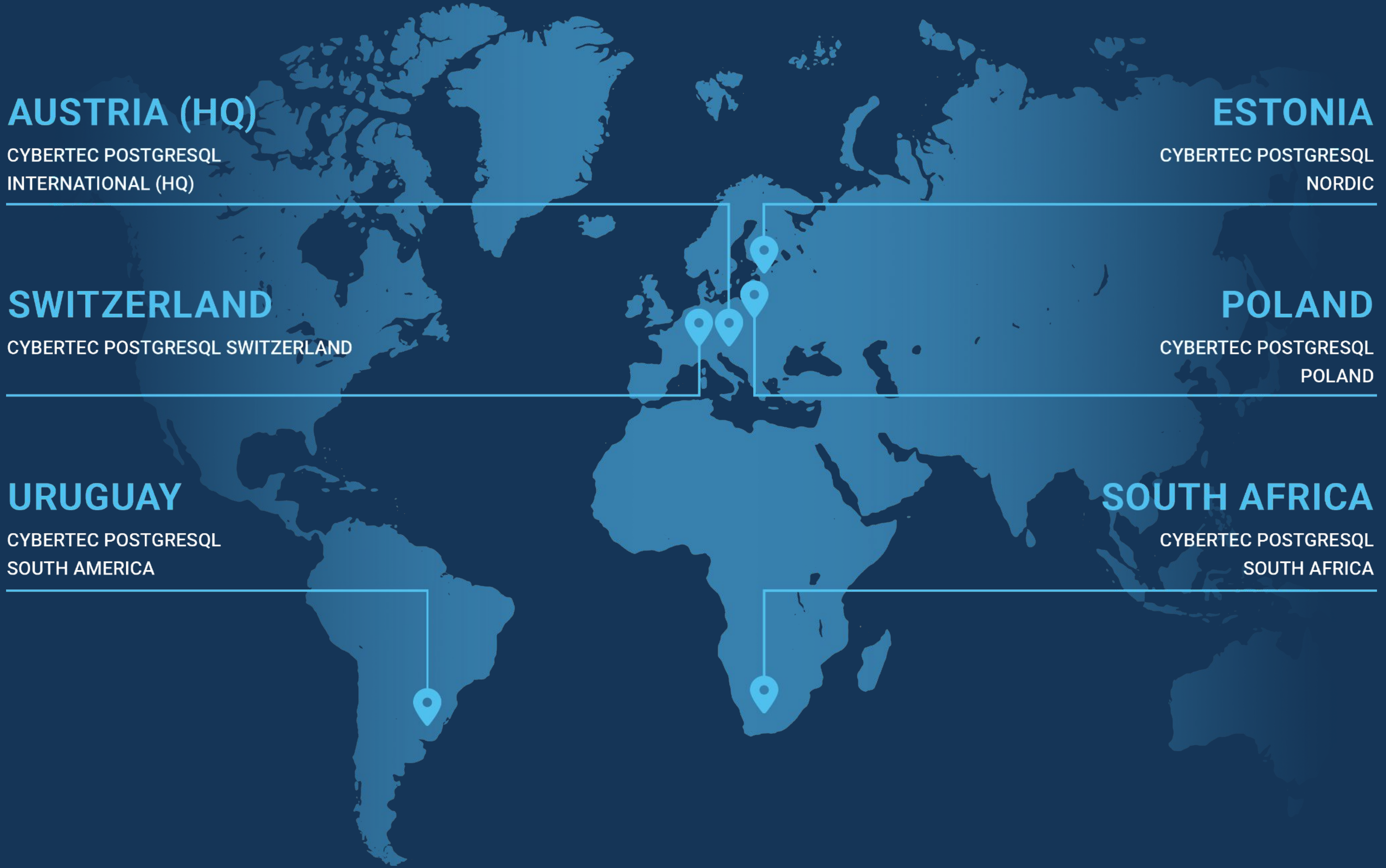
CYBERTEC

POSTGRESQL SERVICES & SUPPORT

POSTGRESQL TOOLING FOR THE COMMUNITY



AUSTRIA (HQ)

CYBERTEC POSTGRESQL
INTERNATIONAL (HQ)

ESTONIA

CYBERTEC POSTGRESQL
NORDIC

SWITZERLAND

CYBERTEC POSTGRESQL SWITZERLAND

POLAND

CYBERTEC POSTGRESQL
POLAND

URUGUAY

CYBERTEC POSTGRESQL
SOUTH AMERICA

SOUTH AFRICA

CYBERTEC POSTGRESQL
SOUTH AFRICA

OPEN SOURCE FOR EVERYONE

THINGS THAT WE FOUND USEFUL

“ OPEN SOURCE MEANS ”

OPEN FOR NEW IDEAS

MANY THINGS IN OUR REPOS

PERFORMANCE AND MONITORING
Tools to monitor and improve performance

AUTOMATION AND ORCHESTRATION
PostgreSQL on Kubernetes and on-prem

ADMINISTRATION AND SECURITY
Manage and secure your databases

https://github.com/cybertec-postgresql/

PERFORMANCE AND MONITORING

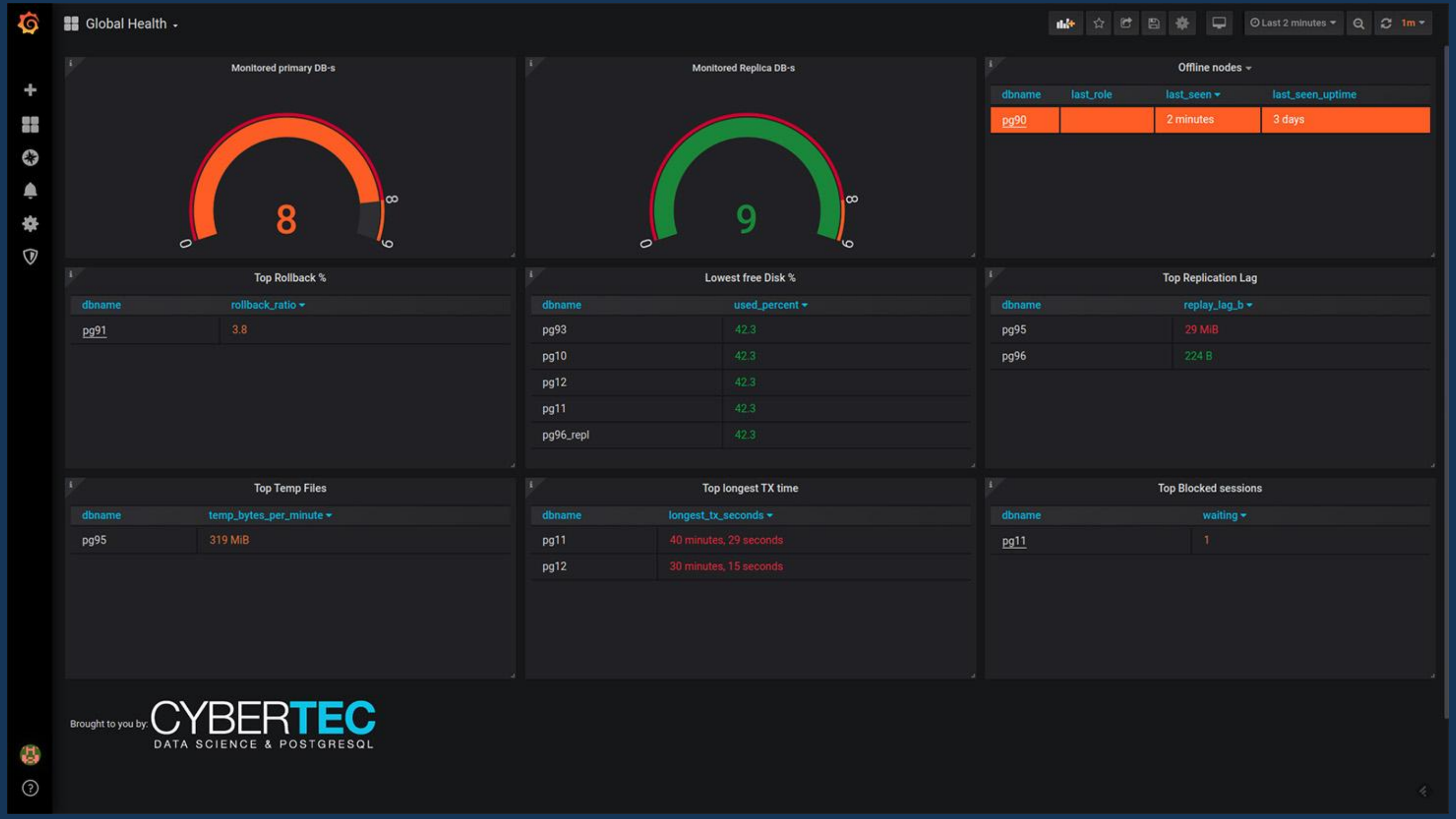
THINGS TO IMPROVE SPEED

PGWATCH2 / 3: ADVANCED MONITORING

- PostgreSQL monitoring at scale
  - We collect EVERY metric PostgreSQL provides
  - Ready made dashboards
  - Automatic service discovery
  - Support all relevant versions

← tested with 10.000 databases

- pgwatch3: Around the corner
  - More modern technology
  - Better at scale
  - More enterprise features



Offline nodes

dbname	last_role	last_seen	last_seen_uptime
pg90		2 minutes	3 days

Top Rollback %

dbname	rollback_ratio
pg91	3.8

Lowest free Disk %

dbname	used_percent
pg93	42.3
pg10	42.3
pg12	42.3
pg11	42.3
pg96_repl	42.3

Top Replication Lag

dbname	replay_lag_b
pg95	29 MiB
pg96	224 B

Top Temp Files

dbname	temp_bytes_per_minute
pg95	319 MiB

Top longest TX time

dbname	longest_tx_seconds
pg11	40 minutes, 29 seconds
pg12	30 minutes, 15 seconds

Top Blocked sessions

dbname	waiting
pg11	1

PG_SHOW_PLANS: LIVE PLAN MONITORING

- “explain” provides execution plans

- But:
    - Which plans are currently running?
    - Plans currently running or not visible
    - How can we analyze running queries?

- pg_show_plans comes to the rescue

← No performance without visibility !

PG_SHOW_PLANS: LIVE PLAN MONITORING

```
testdb=# \x

Expanded display is on.
testdb=# SELECT * FROM pg_show_plans_q;
-[ RECORD 1 ]-----------------------------------------------------------------------------------------
pid   | 11473
level | 0
plan  | Sort  (cost=72.08..74.58 rows=1000 width=80)
      |   Sort Key: pg_show_plans.pid, pg_show_plans.level
      |   ->  Hash Left Join  (cost=2.25..22.25 rows=1000 width=80)
      |         Hash Cond: (pg_show_plans.pid = s.pid)                                                +
      |         Join Filter: (pg_show_plans.level = 0)                                                +
      |         ->  Function Scan on pg_show_plans  (cost=0.00..10.00 rows=1000 width=48)             +
      |         ->  Hash  (cost=1.00..1.00 rows=100 width=44)                                         +
      |               ->  Function Scan on pg_stat_get_activity s  (cost=0.00..1.00 rows=100 width=44)
query | SELECT p.pid, p.level, p.plan, a.query FROM pg_show_plans p                                   +
      |    LEFT JOIN pg_stat_activity a                                                               +
      |    ON p.pid = a.pid AND p.level = 0 ORDER BY p.pid, p.level;
-[ RECORD 2 ]-----------------------------------------------------------------------------------------
pid   | 11517
level | 0
plan  | Function Scan on print_item  (cost=0.25..10.25 rows=1000 width=524)
query | SELECT * FROM print_item(1,20);
-[ RECORD 3 ]-----------------------------------------------------------------------------------------
pid   | 11517
level | 1
plan  | Result  (cost=0.00..0.01 rows=1 width=4)
query |
```

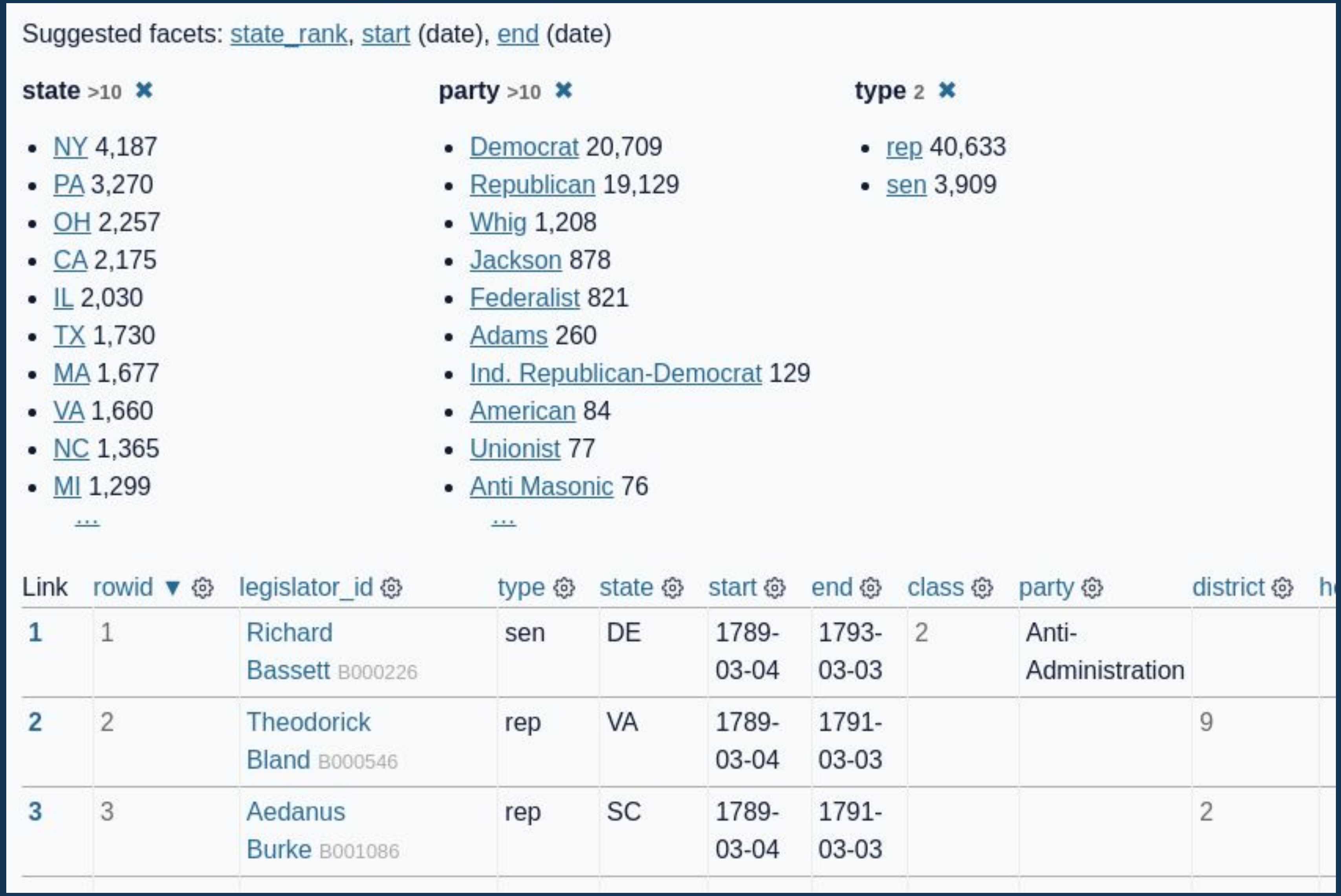
← **Real information in real time**

PGFACETING: SUPER FAST FACETING

- What is faceting in the first place?
- Why is it relevant?

An example of faceting ->

- Usually very expensive
  - Involves expensive counting
  - Slow to implement



Suggested facets: state_rank, start (date), end (date)

state >10 ✖

- NY 4,187
- PA 3,270
- OH 2,257
- CA 2,175
- IL 2,030
- TX 1,730
- MA 1,677
- VA 1,660
- NC 1,365
- MI 1,299

...

party >10 ✖

- Democrat 20,709
- Republican 19,129
- Whig 1,208
- Jackson 878
- Federalist 821
- Adams 260
- Ind. Republican-Democrat 129
- American 84
- Unionist 77
- Anti Masonic 76

...

type 2 ✖

- rep 40,633
- sen 3,909

Link	rowid ▼	legislator_id	type	state	start	end	class	party	district	h
1	1	Richard Bassett B000226	sen	DE	1789-03-04	1793-03-03	2	Anti-Administration		
2	2	Theodorick Bland B000546	rep	VA	1789-03-04	1791-03-03			9	
3	3	Aedanus Burke B001086	rep	SC	1789-03-04	1791-03-03			2	

PGFACETING: SUPER FAST FACETING

- Super fast implementation using “roaring bitmaps”
- Experiment with 100.000.000 rows
- Plain SQL takes 4 min 42 seconds

```
postgres=# SELECT facet_name, count(distinct facet_value), sum(cardinality)
           FROM faceting.count_results('documents'::regclass,
                filters => array[row('category_id', 24)]::faceting.facet_filter[])
           GROUP BY 1;
 facet_name | count |   sum
------------+-------+----------
 created    |   154 | 60812252
 finished   |   154 | 60812252
 size       |     7 | 60812252
 type       |     8 | 60812252
(4 rows)

 Time: 155.228 ms
```

https://github.com/cybertec-postgresql/pgfaceting

AUTOMATION AND ORCHESTRATION

POSTGRESQL ON KUBERNETES

AND ON-PREMISE

POSTGRESQL OPERATOR FOR KUBERNETES

- Fully functional PostgreSQL Operator for …
  - Kubernetes / OpenShift / Rancher
  - RedHat certified package available !

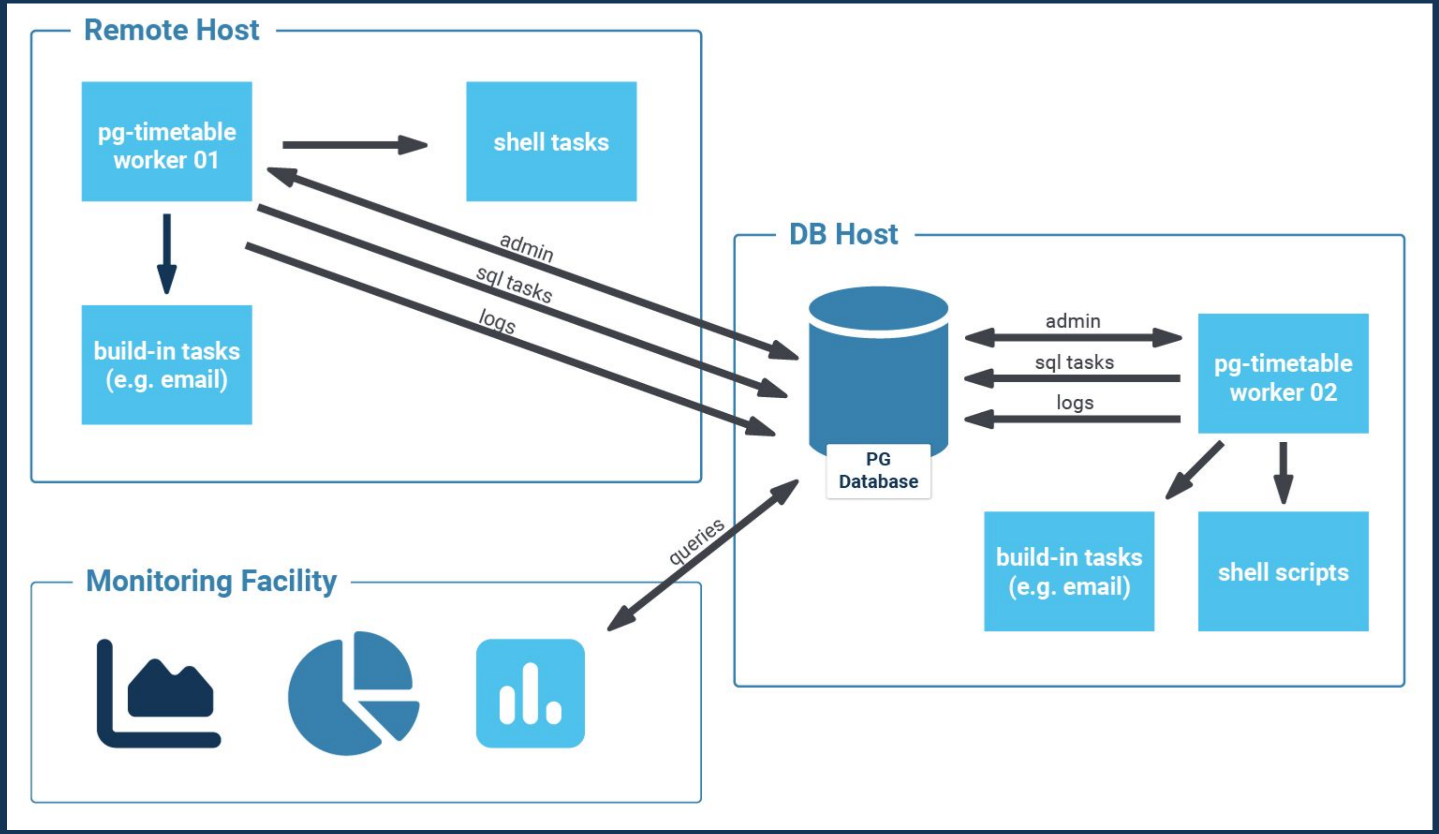
- Substantial improvements over the Zalando operator
  - Faster development cycle
  - Made for more generic needs

- Full support available provided by us
- Soon available for “Multi-Site Kubernetes”

https://github.com/cybertec-postgresql/CYBERTEC-pg-operator

PG_TIMETABLE: SCHEDULING DONE PROPERLY

- Fully features PostgreSQL scheduler
- NO server side stuff needed (processes, modules, etc).
- Single, easy to deploy binary
- Can run:
  - Single tasks
  - Chains of tasks
  - SQL, builtin and Shell tasks
- Support:
  - Non-overlapping execution
  - Async execution (“suicide jobs”)



https://github.com/cybertec-postgresql/pg_timetable

ADMINISTRATION AND SECURITY

OPERATION EXCELLENCE

PG_PERMISSIONS: AUDIT AND SECURITY

- Security audits become more frequent
- Security does matter
- How can we …
  - Compare: “Reality” vs “desired state”
  - See all permissions at one glance

- pg_permissions does all of that and more

https://github.com/cybertec-postgresql/pg_permissions

PG_PERMISSIONS: AUDIT AND SECURITY

```
test=# SELECT * FROM all_permissions WHERE role_name NOT LIKE 'pg%';
-[ RECORD 1 ]------------------------------------
object_type | TABLE
role_name   | joe
schema_name | columnar_internal
object_name | options
column_name |
permission  | SELECT
granted     | f
-[ RECORD 2 ]------------------------------------
object_type | TABLE
role_name   | joe
schema_name | columnar_internal
object_name | options
column_name |
permission  | INSERT
granted     | f
…
```

https://github.com/cybertec-postgresql/pg_permissions

PG_PERMISSIONS: AUDIT AND SECURITY

```
INSERT INTO public.permission_target
   (role_name, permissions,
    object_type, schema_name, object_name)
VALUES
   ('appuser', '{USAGE}',
    'SEQUENCE', 'appschema', 'appseq');

SELECT * FROM public.permission_diffs();
 missing | role_name | object_type | schema_name | object_name | column_name | permission
---------+-----------+-------------+-------------+-------------+-------------+------------
 f       | laurenz   | VIEW        | appschema   | appview     |             | SELECT
 t       | appuser   | TABLE       | appschema   | apptable    |             | DELETE
(2 rows)
```

https://github.com/cybertec-postgresql/pg_permissions

PG_SQUEEZE: ENDING TABLE BLOAT

- Shrink table WITHOUT excessive locking
  - Just a short lock at the end

- Shrink tables when VACUUM cannot help anymore
- Especially useful when facing “hyper bloat”
  - For example: 1 GB -> 1 TB (no way to fix with normal VACUUM)

https://github.com/cybertec-postgresql/pg_squeeze

PG_SQEEZE: AD HOC ACTION

```
CREATE EXTENSION pg_squeeze;

SELECT squeeze.squeeze_table('public', ‘t_test’);
```

Shrinking on demand

Be prepared for potential failures
It can happen by design in some cases

PG_SQEEZE: SCHEDULED ACTION

```
INSERT INTO squeeze.tables (
    tabschema,
    tabname,
    schedule,
    free_space_extra,
    vacuum_max_age,
    max_retry)
VALUES (
    'public',
    't_test',
    ('{30}', '{22}', NULL, NULL, '{3, 5}'),
    30,
    '2 hours',
    2
);
```

schedule shrinking

THERE IS A LOT MORE

AND MORE IS TO COME

ANY QUESTIONS?

FEEL FREE TO ASK

HANS-JÜRGEN SCHÖNIG

CEO & FOUNDER

EMAIL

hs@cybertec-postgresql.com

PHONE

+43 2622 930 22 - 666

WEB

www.cybertec-postgresql.com