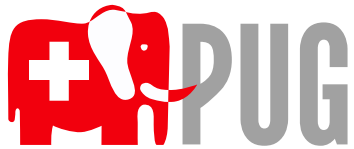


Pattern matching in the Unicode space

Johannes Graën
johannes.graen@uzh.ch

University of Zurich[UZH]

To boldly go where no one has gone before

Pattern matching in the Unicode space

Johannes Graën

Friday 28th June, 2024

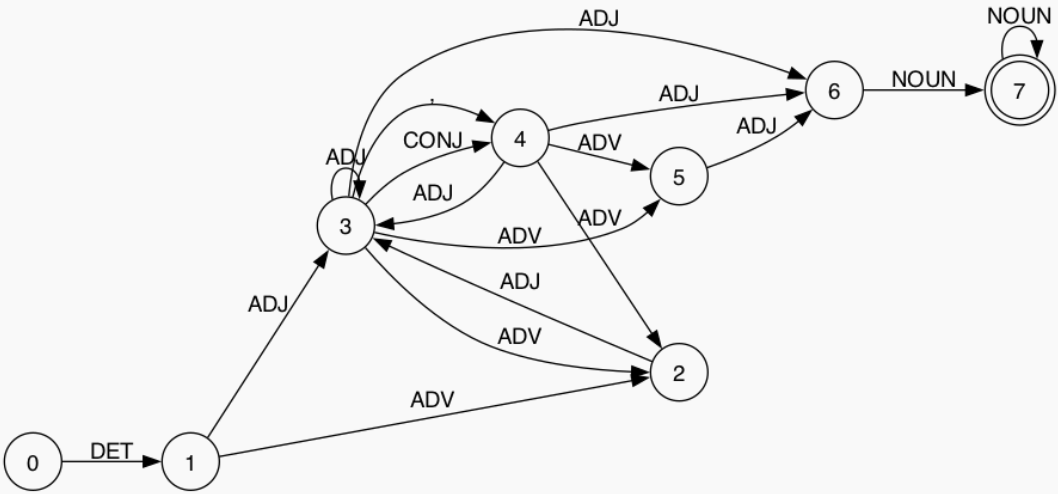
Background

Pattern matching on attributes of word sequences (among others)

Example
Find a noun phrase that consists of the following sequence:

1. one determiner (article) – `DET`
2. at least two times
    1. an optional adverb – `ADV`
    2. one adjective – `ADJ`
    3. optionally a comma or a conjunction – `CONJ`
       (unless in front of a noun)
3. at least one noun – `NOUN`

Representation as automaton (DCG)



Implementation as recursive CTE

... works, but performance is rather underwhelming

```
WITH RECURSIVE transition (src, trg) AS (
  VALUES (1, 2), (1, 3), (2, 3), (3, 3), (3, 4), (3, 5), (3, 6), ...
), search AS (
  SELECT list.position, 1 AS state, ...
  FROM list
  WHERE upos = 'DET'
UNION ALL
  SELECT list.position, t.trg AS state, ...
  FROM search s
  JOIN transition t ON t.src = s.state
  JOIN list ON list.position = s.position + 1
  WHERE
    (list.upos = 'ADJ' AND ((t.src, t.trg) IN ((1, 3), (2, 3), ...))
    OR (list.upos = 'ADV' AND ((t.src, t.trg) IN ((1, 2), (3, 2), ...))
    OR ...
) ...
```

Better solution? Can we use regular expressions?

- regular expressions can do everything we want
- but we need to represent data as strings
- ... which shouldn't be a problem as Unicode has more than 1m code points

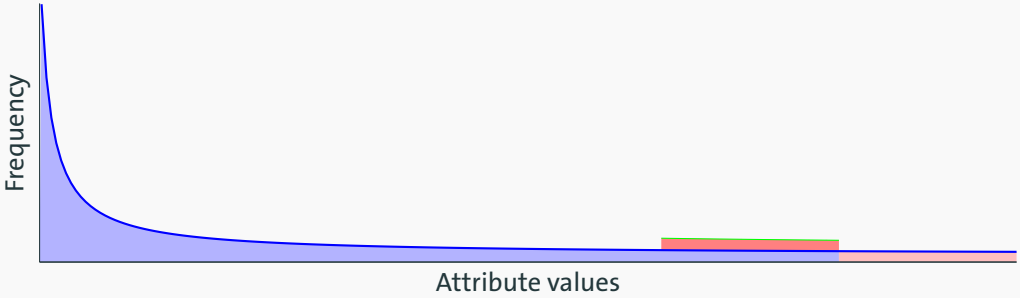
Approach

1. map attributes to code points
2. construct strings as representations of attributes
3. compile patterns to regular expressions

Unicode

- 17 planes of 2^{16} code points each
- minus 2048 surrogates
- $\Rightarrow \approx$ 1.1m code points
- different Unicode Transformation Formats (UTF) – in Postgres only UTF-8:
  - 1 Byte for CPs from U+0000 to U+007F
    – 128 CPs $\approx$ 0,012 %
  - 2 Bytes for CPs from U+0080 to U+07FF
    – 1 920 CPs $\approx$ 0,17 %
  - 3 Bytes for CPs from U+0800 to U+D7FF and from U+E000 to U+FFFF
    – 61 440 CPs $\approx$ 5,5 %
  - 4 Bytes for CPs from U+10000 to U+10FFFF
    – 1 048 576 CPs $\approx$ 94,3 %

Feature distributions follows Zipf's law



- order values by frequency and assign code points
- skip characters used by regular expressions (otherwise we need some escaping)
- reserve one character for word boundary
- ... and another one for NULL values
- if we outrun the space, reuse upper code point space (red part)
  => lossy representation

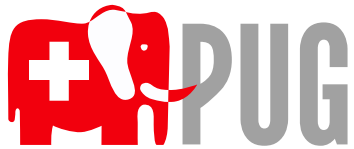
Pattern matching

```
SELECT id
FROM cp_rep
CROSS JOIN (
  SELECT format(
    E'.%1$s.\n((.%2$s.\n).%3$s.\n((.%5$s|%6$s.).\n)?){2,}(.%4$s.)+',
  (SELECT chr(cp) FROM map_upos_cp WHERE upos = 'DET'),          -- $1
  (SELECT chr(cp) FROM map_upos_cp WHERE upos = 'ADV'),          -- $2
  (SELECT chr(cp) FROM map_upos_cp WHERE upos = 'ADJ'),          -- $3
  (SELECT chr(cp) FROM map_upos_cp WHERE upos = 'NOUN'),         -- $4
  (SELECT chr(cp) FROM map_upos_cp WHERE upos = 'CCONJ'),        -- $5
  (SELECT chr(cp) FROM map_form_cp JOIN form USING (form_id)     -- $6
    WHERE form = ',')
  ) AS r
) x
WHERE s ~ r;
```

⇒ Parallel sequential scan on precalculated strings

Some results

- a highly intelligent, very sophisticated animal
- a very sweet, very cute, very dead man
- the so various only wild boars
- the sometimes thrilling, sometimes sordid, always mysterious world
- a very regular almost parallel fashion
- the most important, most talented, most interesting and most extraordinary person



YUGABYTEDB: A Postgres fork to scale horizontally

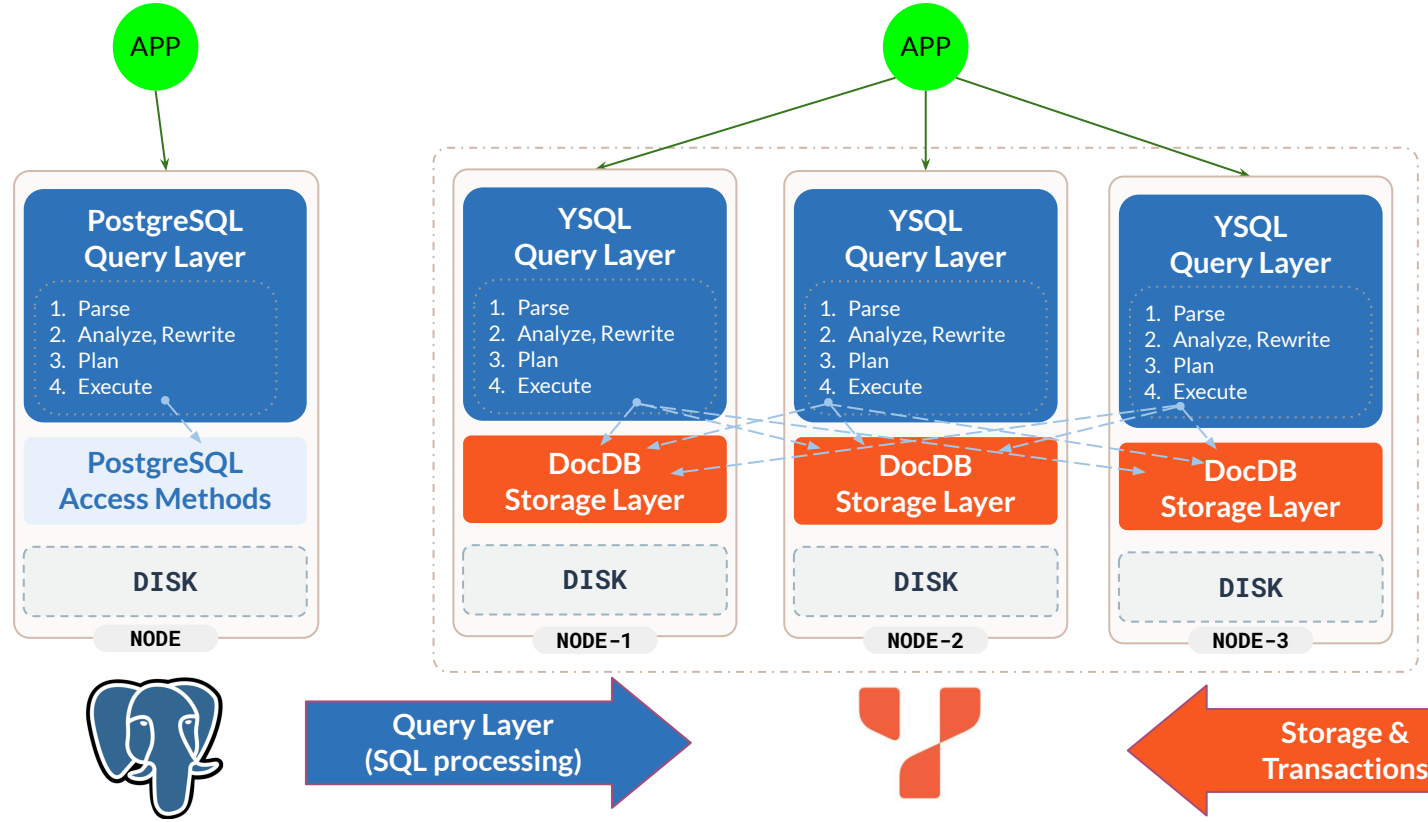
Franck Pachot
franck@pachot.net

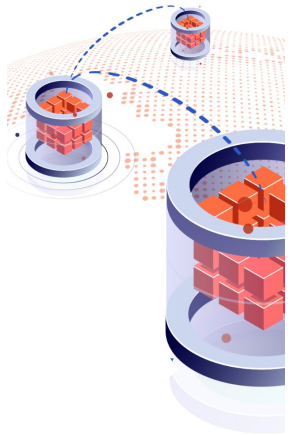
YugabyteDB: a PostgreSQL fork with Horizontal Scalability

yugabyteDB

@FranckPachot

Monolithic PostgreSQL ⋈ Distributed YugabyteDB





Shards are LSM-Trees (RocksDB with read optimizations)

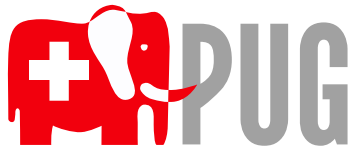
```
yugabyte=# explain ( analyze, dist, debug, costs off, summary off)
           select distinct aid from pgbench_accounts order by aid;
```

```
                                                        QUERY PLAN
-----------------------------------------------------------------------------------------------------------
Unique (actual time=1.088..73.485 rows=100000 loops=1)
   ->  Index Only Scan using idx on pgbench_accounts
        (actual time=1.087..31.300  rows=100000 loops=1)
         Heap Fetches: 0
         Storage Index Read Requests: 98
         Storage Index Read Execution Time: 4.545 ms
         Metric rocksdb_number_db_seek: 98.000
         Metric rocksdb_number_db_next: 100097.000
         Metric rocksdb_number_db_seek_found: 98.000
         Metric rocksdb_number_db_next_found: 100096.000
         Metric rocksdb_iter_bytes_read: 4249346.000
         Metric ql_read_latency: sum: 38190.000, count:  98.000
```

YugabyteDB: a PostgreSQL fork with Horizontal Scalability

yugabyteDB

@FranckPachot



TDE is coming in…

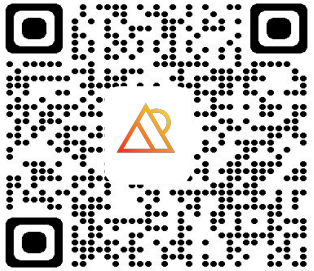
Kai Wagner
kai.wagner@percona.com

PERCONA

TDE is comin in...

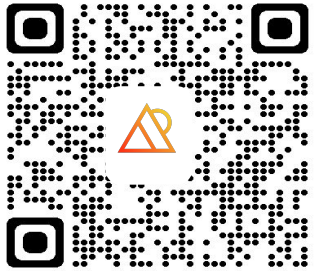
Kai Wagner <kai.wagner@percona.com>

Why do we need encryption?



- Data Security
- Privacy Protection
- Prevention of Data Breaches
- Compliance

Hot topic within the community for years



While almost everyone agrees that encryption is good to have, not everyone believes this should be part of the PG core.

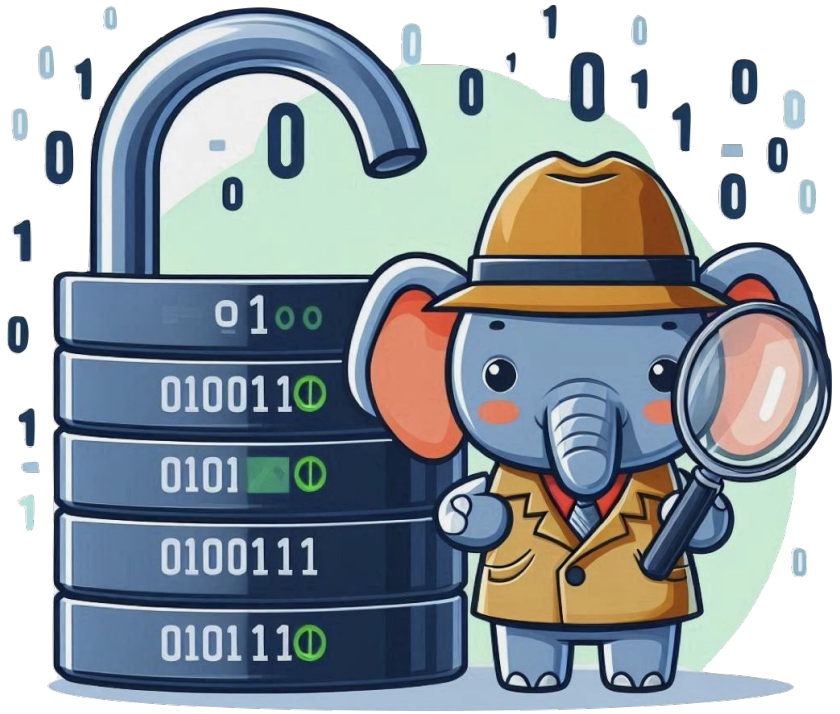
<Quote by Kai>

pg_tde

1. Encrypted access method
2. Extension instead of core
3. Open Source (no strings attached)
4. Encryption for everyone & everywhere
5. Flexible
    a. No vendor lock-in

So....what's encrypted in the extension

- User data in tables
  - including TOAST tables, that are created using the extension.
- Write-Ahead Log (WAL) data for tables created using the extension
- Temporary tables created during the database operation for data tables created using the extension



So....what's not encrypted?

- No index encryption

Index encryption is coming!

2 solutions in 1

In short - if you don't need or use indexes use the extension, otherwise go with the patch.

A

```
CREATE TABLE my_table (id SERIAL,
pii_data VARCHAR(32), PRIMARY
KEY(id)) USING pg_tde;
```

This encrypts only the table.

Also allows WAL encryption
(optional) for the whole cluster

Extension

Everything in option A and more

B

```
CREATE TABLE my_table (id SERIAL,
pii_data VARCHAR(32), PRIMARY
KEY(id)) USING pg_tde_full;
```

This encrypts the table and everything related.

System tables for now stay not encrypted.
Potential future increment.

If you know better. Let us know.
We will be talking to UX too!

Extension + Patch

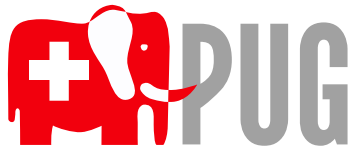
Superior features of pg_tde

- Multitenancy
  - Separate key per database
- Key-Management
- Key-Rotation
- Table level granularity
- Vanilla Support or binary compatibility (drop-in)

What's next?

- We're working closely with upstream, to get needed SMGR changes into PostgreSQL 18.
  - If this succeeds, there is no need for a patch anymore
  - "Only" XLog would be left and we would focus on getting this change, also into the core
- End Goal - Full Encryption through an extension, without the need to patch PostgreSQL - Simply works with Vanilla/Upstream.

Thank You!



The Well-Tempered Elephant

Gianni Ciolli

EDB
Postgres® for the AI Generation

The Well-Tempered Elephant

Gianni CIolli
Global VP, Practice Lead High Availability
PGDay.CH, 27-28 June 2024

The Well-Tempered Clavier

- Published by Johann Sebastian Bach
- The **most important** piano music
- Composed by two books: 1722 and 1742
- In total, 48 Prelude and Fugue pieces
- 48 = 2 (books) x 12 (keys) x 2 (modes)

The fugues in the Well-Tempered Clavier

- Fugues follow a precise format and can be analyzed
- Kyle Rother (University of Cape Town)
  transcribed the 48 fugues in **digital format** (Lilypond)
- We import these fugues into PostgreSQL
  - `Lilypond`: compile sources into MIDI files
  - `mftext`: dump notes from MIDI files to stdout
  - COPY: import notes in a table

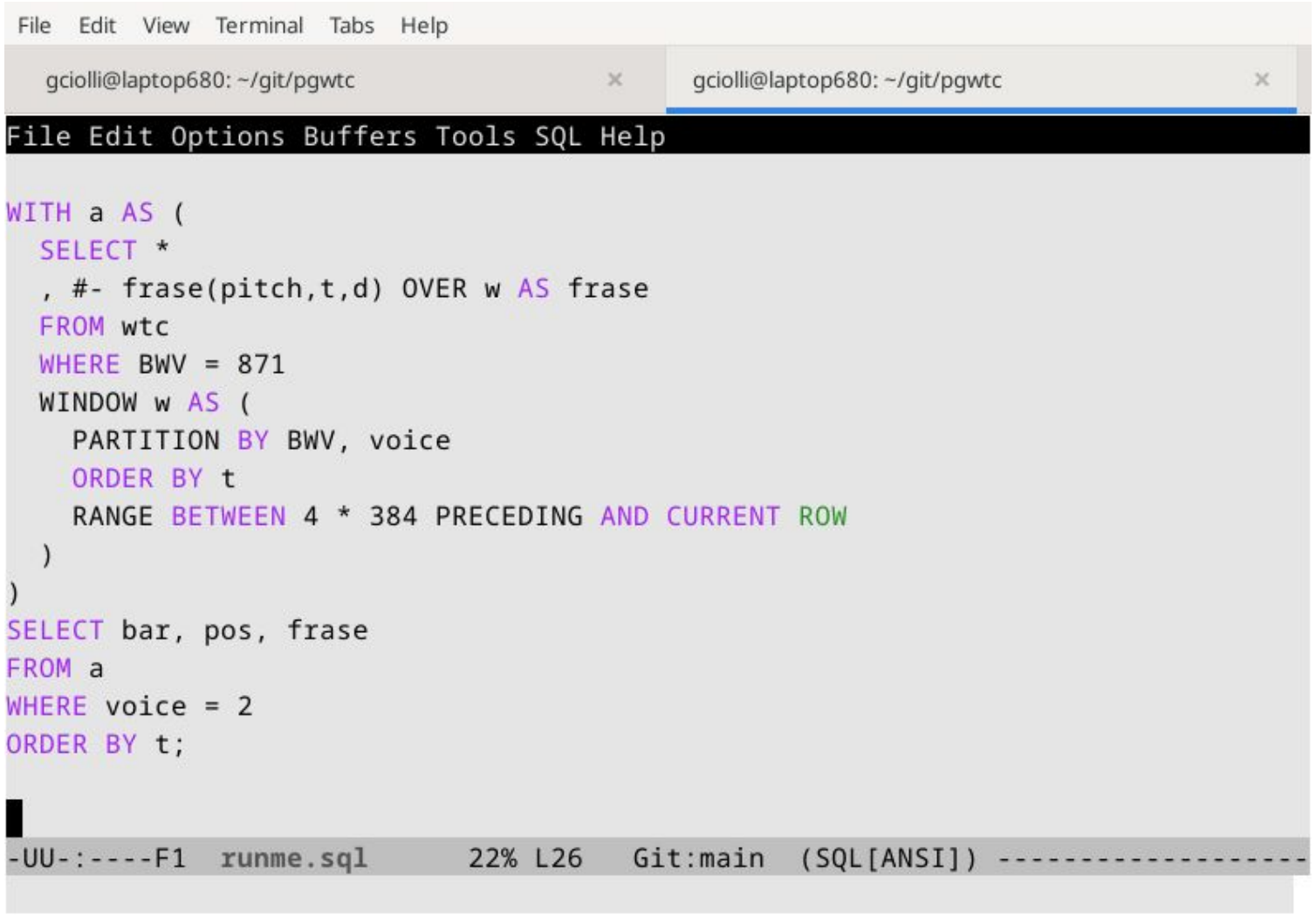
The Well-Tempered Elephant, in short: `pgwtc`

- Use PostgreSQL for the analysis of fugues
- The WTC is the obvious starting point,
  but `pgwtc` can be used for any fugue
- Load notes in a Postgres table
  - A fugue is composed by 2-5 voices
  - A voice is a sequence of notes and pauses
  - Total: 51045 notes (after data cleaning)

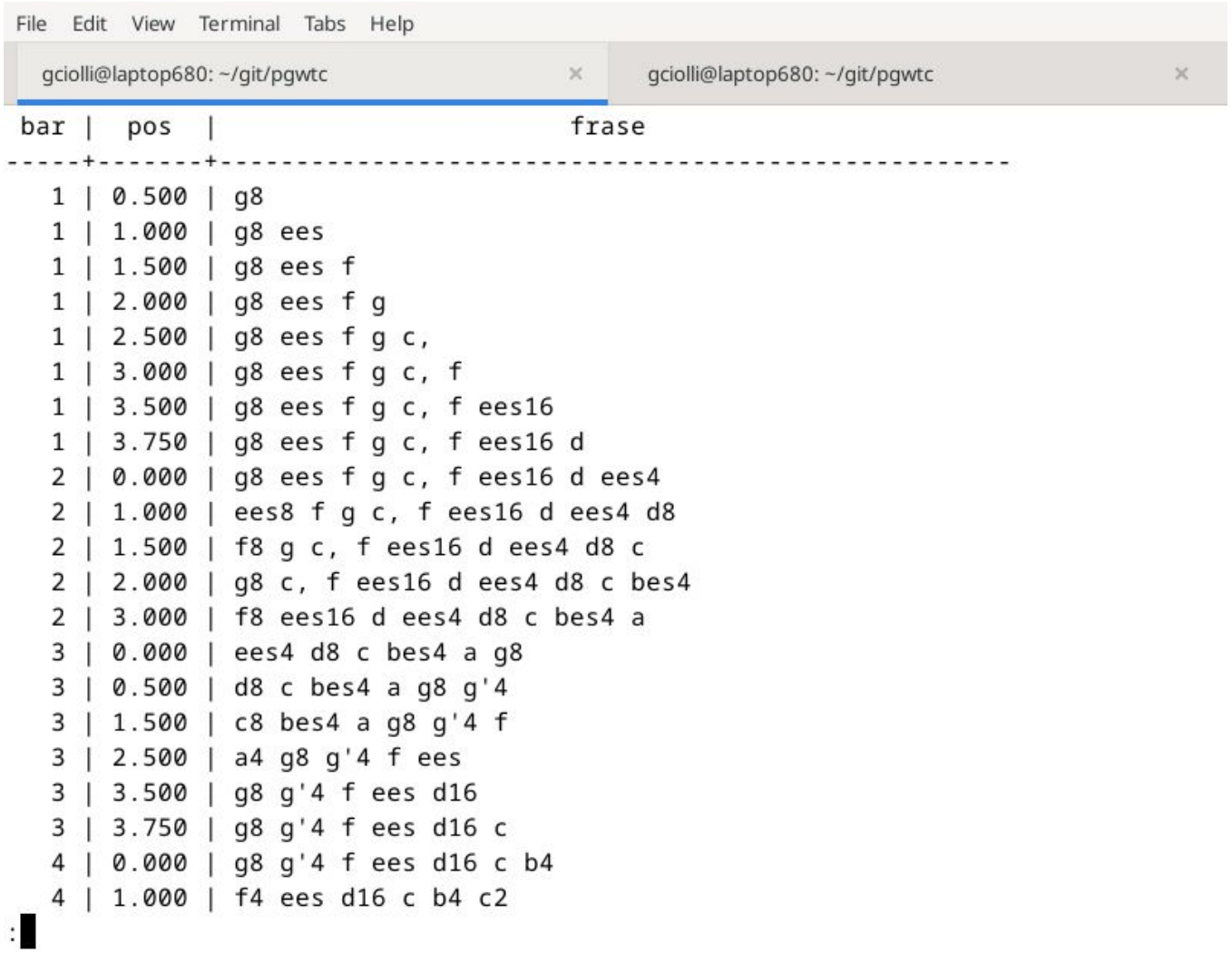
Why PostgreSQL?

- **COPY** to easily **load** text data into a table
- **Custom aggregates** to **clean** the data
- **Window functions** to **aggregate** sequences of notes
- **Custom operators** to **display** sequences of notes and **recognise** themes as matching sequences of notes
- **Extensions** to facilitate **reuse** of the above tools for the analysis of other fugues

Screenshot #1: Example of query

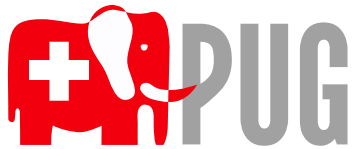


Screenshot #2: Example of output



Thank you!

(and thank J. S. Bach too)



PostgreSQL in the snow send the right athletes to the finals

Andreas Gruhler

PostgreSQL in the snow

Lightning talk 28.06.2024

Andreas Gruhler

Share | Notes | Edit | Print

The setting, slopestyle contests

This is not a presentation 🤗

https://myheats-demo.p0c.ch

The process

- Judges submit scores for athletes on paper
- Event admins transfer scores to Excel
- Excel is synced (network!) to office
- Office shuffles and sorts rows
- New heats (list of athletes) sent to judges

The output/reference

Bündnermeisterschaften der Schneesportlehrer*innen 2023

Slopestyle Snowboard

Schlussklassement

					Qualifikation										Finaldurchgang				Endnote
					Qualifikation, Lauf 1				Qualifikation, Lauf 2				bessere	*Rang*					
					Note 1 Kathrina Erdin	Note 2 Christian Fallegger	Note 3 Emilie Benz	Wertung	Note 1 Kathrina Erdin	Note 2 Christian Fallegger	Note 3 Emilie Benz	Wertung	Wertung		Note 1 Kathrina Erdin	Note 2 Christian Fallegger	Note 3 Emilie Benz	Wertung	
Rang	**Nr**	**Name Vorname**	**Jg**	**Schule**															
Damen																			
	2	Marti Bettina	1990	Klosters	8.1	8.1	7.8	24.0	7.7	7.8	7.8	23.3	**24.0**		8.0	7.9	7.8	**23.7**	**47.7**
	8	Chudoba Svenja	1990	Lenzerheide	8.0	8.2	7.9	24.1	7.1	6.7	6.9	20.7	**24.1**		7.8	7.8	7.6	**23.2**	**47.3**
	6	Tempini Melissa	1996	Lenzerheide	7.1	7.3	7.0	21.4	7.4	7.3	7.2	21.9	**21.9**		7.4	7.2	7.1	**21.7**	**43.6**
	10	Keller Caroline	1995	Lenzerheide	7.2	6.8	6.9	20.9	7.3	7.2	7.3	21.8	**21.8**		7.3	7.1	7.3	**21.7**	**43.5**
	5	Meandzija Mona	1994	Savognin	6.2	6.5	6.5	19.2	7.3	7.1	7.1	21.5	**21.5**		7.5	7.3	7.2	**22.0**	**43.5**
	7	Henggeler Ladina	1995	Klosters	7.0	6.8	7.1	20.9	7.3	7.0	7.1	21.4	**21.4**		7.0	6.9	6.9	**20.8**	**42.2**
	1	Reusser Eliane	1990	Klosters	6.8	6.8	6.8	20.4	7.1	7.1	7.0	21.2	**21.2**					**0.0**	**0.0**
	9	Fisler Lara	2003	Savognin	6.7	6.6	6.7	20.0	7.0	6.6	6.6	20.2	**20.2**					**0.0**	**0.0**
	4	Clemente Lorena	1997	St. Moritz	5.8	5.6	6.0	17.4	7.0	6.2	6.7	19.9	**19.9**					**0.0**	**0.0**
Herren																			
	29	Spescha Ciril	1994	Brigels	7.9	7.8	7.3	23.0	8.5	8.3	8.3	25.1	**25.1**		8.3	8.5	8.4	**25.2**	**50.3**
	25	Fischediek Tillmann	1999	Savognin	8.4	8.4	8.1	24.9	7.2	6.7	7.2	21.1	**24.9**		8.2	8.3	8.3	**24.8**	**49.7**
	30	Hager Robin	1994	Davos	8.8	8.4	8.5	25.7	6.5	6.7	6.7	19.9	**25.7**		7.5	7.9	7.9	**23.3**	**49.0**
	38	Coester Juri	1994	Davos	8.0	8.0	8.1	24.1	8.0	8.0	7.9	23.9	**24.1**		7.9	8.0	7.9	**23.8**	**47.9**

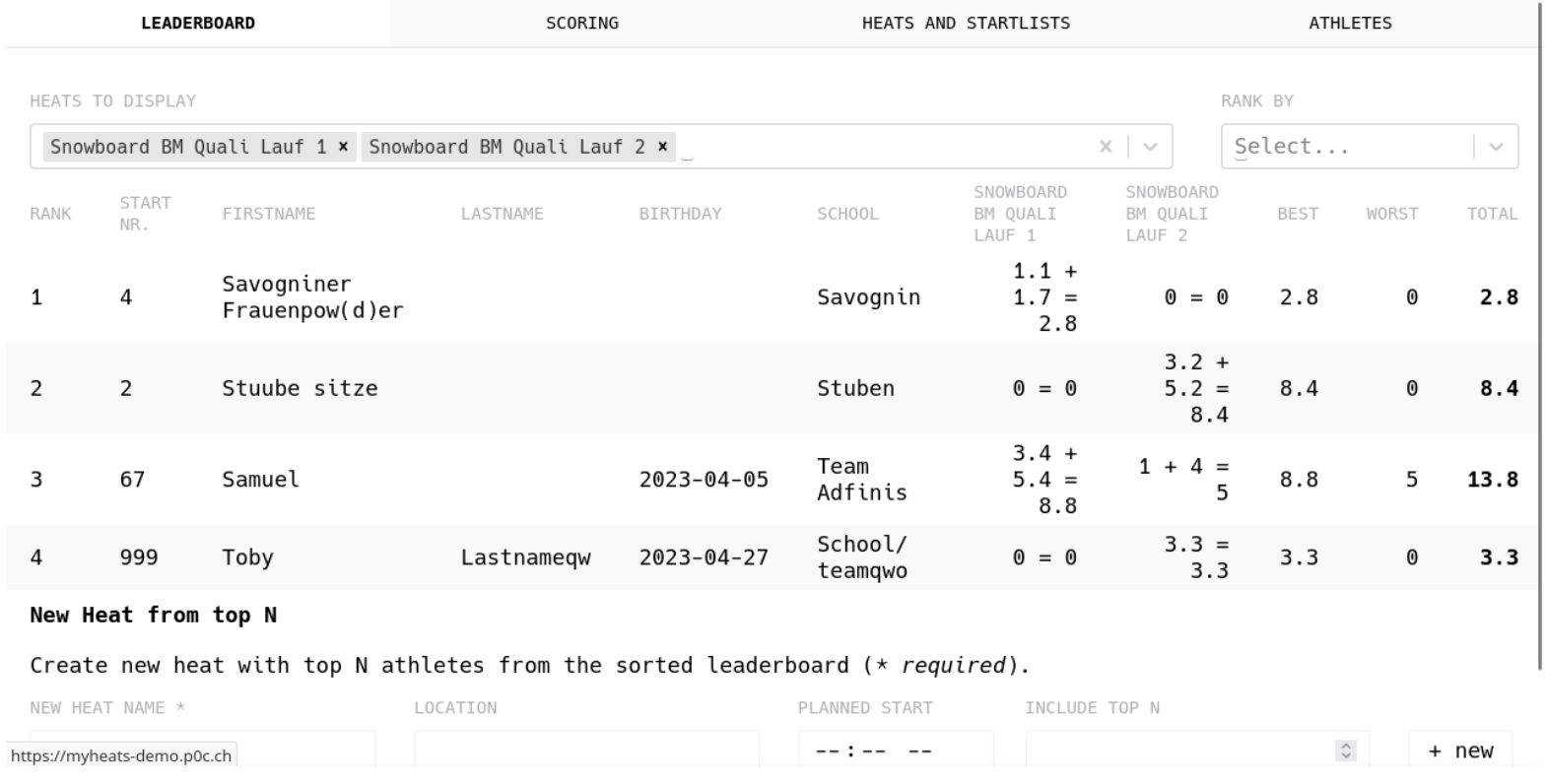
The problem

- Organizer: How to *not* send the wrong athletes to the finals?
- Athlete/Audience: No live stats, how did I do in my run?

Inefficient scoring and reporting process in slopestyle contests:

- Errors pile up quickly in additional heats
- Different medias (paper -> xls)
- Existing scoring platforms are complex (expensive, privacy concerns)

One solution



LEADERBOARD | SCORING | HEATS AND STARTLISTS | ATHLETES

HEATS TO DISPLAY: Snowboard BM Quali Lauf 1 × Snowboard BM Quali Lauf 2 ×

RANK BY: Select...

RANK	START NR.	FIRSTNAME	LASTNAME	BIRTHDAY	SCHOOL	SNOWBOARD BM QUALI LAUF 1	SNOWBOARD BM QUALI LAUF 2	BEST	WORST	TOTAL
1	4	Savogniner Frauenpow(d)er			Savognin	1.1 + 1.7 = 2.8	0 = 0	2.8	0	**2.8**
2	2	Stuube sitze			Stuben	0 = 0	3.2 + 5.2 = 8.4	8.4	0	**8.4**
3	67	Samuel		2023-04-05	Team Adfinis	3.4 + 5.4 = 8.8	1 + 4 = 5	8.8	5	**13.8**
4	999	Toby	Lastnameqw	2023-04-27	School/ teamqwo	0 = 0	3.3 = 3.3	3.3	0	**3.3**

New Heat from top N

Create new heat with top N athletes from the sorted leaderboard (* *required*).

NEW HEAT NAME * | LOCATION | PLANNED START --:-- -- | INCLUDE TOP N | + new

https://myheats-demo.p0c.ch

Use a database (PostgreSQL, Supabase)

The requirements

- Mobile, screens are expensive, phones are ubiquitous
- Excel compatible, it’s just tabular score data
- Local, keep in control of your data (privacy)
- Reasonably secure and performant

Realtime feature with PostgreSQL publications

Status: open
Priority: 0
Public link
Back to the board

Swimlane: Default swimlane
Column: Backlog
Position: 7

Assignee: not assigned
Creator: Andreas

Started:
Created: 03/21/2024 20:42
Modified: 03/21/2024 20:42
Moved: 03/21/2024 20:42

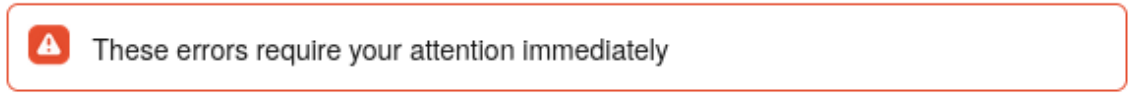
postgresql supabase realtime publication

▼ Description

Can the Supabase realtime feature be coded/replaced only using PostgreSQL publication?

It's seems to be possible using the `porsager/postgres` lib.

User education



Hello agruhl@gmx.ch,

You currently have project(s) that trigger warnings in our security advisor. This is a weekly reminder to review these issues.

Report created [14 May 2024].

Project: MyHeats Demo

ID: aaxkgqazjhwumoljibld

1 error(s)

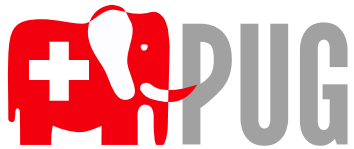
View Security Advisor

Security & performance advisors

```
 ```sql
 -- Summarize all scores of a heat
-create or replace view score_summary as
+create or replace view score_summary
+  with (security_invoker=on)
+  as
   select a.id as athlete_id, s.heat as heat_id, SUM(s.score) as score_summary
   from scores s
   join athletes as a on a.id = s.athlete
@@ -47,12 +49,13 @@ create or replace function distinct_startlist(heat_ids numeric[])
 returns table(id bigint, athlete bigint, nr bigint, firstname text, lastname text,
 birthday date, school text)
 language plpgsql
+set search_path = ''
 as $$
 begin
   return query
   select distinct on (a.id) s.id, a.id, a.nr, a.firstname, a.lastname, a.birthday, a.school
-  from startlist as s
-  join athletes as a on s.athlete = a.id
+  from public.startlist as s
+  join public.athletes as a on s.athlete = a.id
   where s.heat = any (heat_ids);
 end;
 $$;
@@ -62,10 +65,11 @@ Add a trigger for connecting users with judges:
 ```sql
 -- https://supabase.com/docs/guides/auth/managing-user-data#using-triggers
 -- inserts a row into public.judges
-create function public.handle_new_judge()
+--drop trigger if exists on_auth_user_created on auth.users;
+create or replace function public.handle_new_judge()
 returns trigger
 language plpgsql
-security definer set search_path = public
+security definer set search_path = ''
 as $$
 begin
   insert into public.judges (id)
```

Hack on

- All in a box (Raspberry Pi)
  - Realtime feature with PostgreSQL publications
  - Javascript magic link authentication
- Export/import



A Song of Ice and Fire

Pavlo Golub
pavlo.golub@cybertec.at

A Song of Ice and Fire

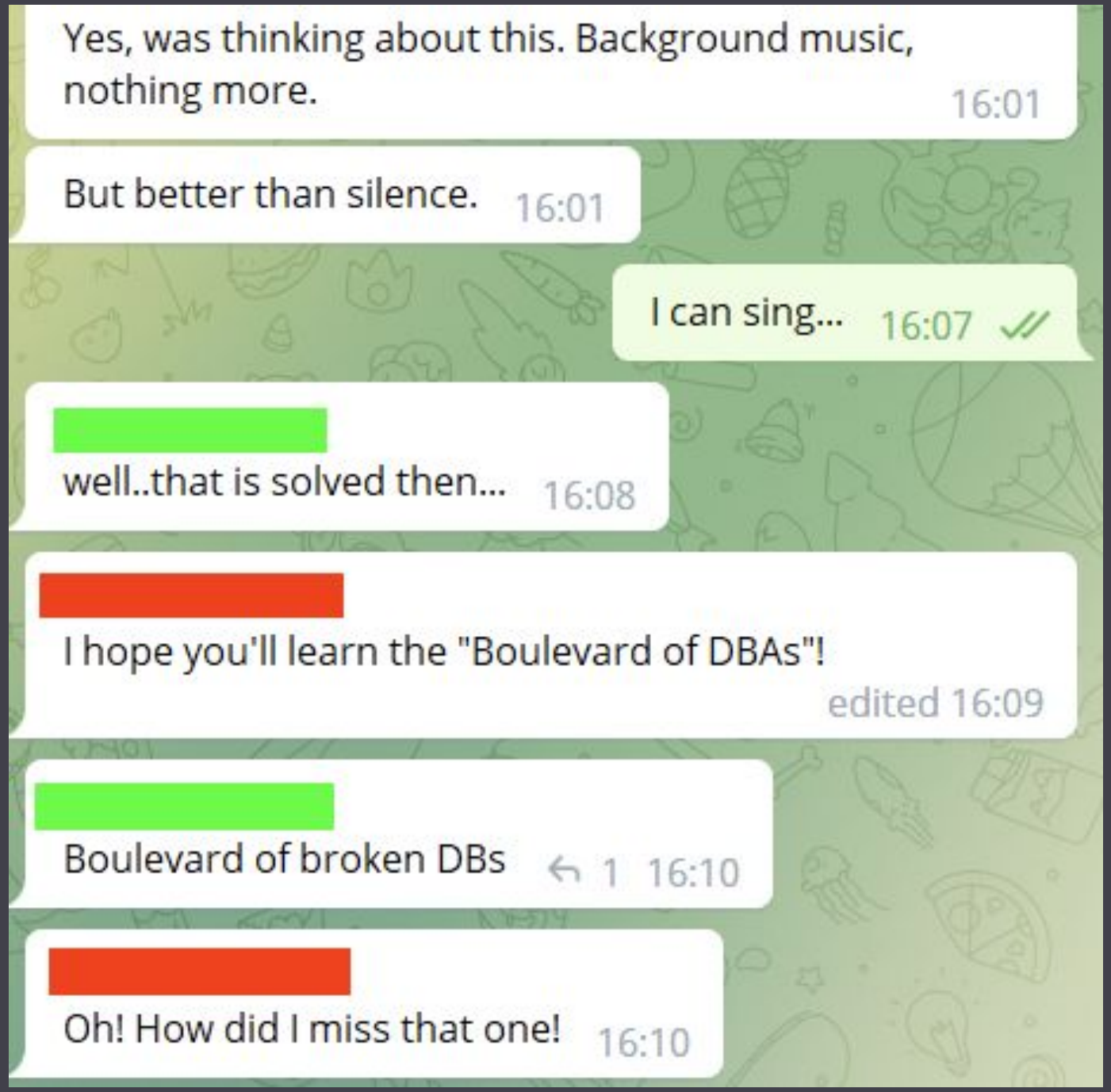
Pavlo Golub

Senior Database Consultant

pavlo.golub@cybertec.at

@PavloGolub

CYBERTEC
The PostgreSQL Database Company


Yes, was thinking about this. Background music, nothing more. 16:01
But better than silence. 16:01
I can sing... 16:07
well..that is solved then... 16:08
I hope you'll learn the "Boulevard of DBAs"! edited 16:09
Boulevard of broken DBs 1 16:10
Oh! How did I miss that one! 16:10

Nothing Compares To VACUUM/The Ballad of Bloat

2,103 views Premiered on 21 Jan 2020 PostgreSQL VACUUM Song
Singer: Gabriel Cafa ...more

Explain Analyze (Feliz Navidad cover)

934 views 26 May 2021 lyrics by Michael Christofides inspired by this blog post
https://www.ongres.com/blog/explain_a... ...more

let's make it professional way:

```
create extension fuzzystrmatch;

SELECT word from pg_get_keywords() WHERE
difference('dreams', word) >= 2;
```

16:38

Boulevard of broken [plans | trims | routines | drops | froms | grants | groups] :)

edited 16:39

fuzzystrmatch

The fuzzystrmatch module provides several functions to determine similarities and distance between strings:

- Soundex
- Levenshtein
- Metaphone & Double Metaphone

pg_get_keywords ()

- Returns a set of records describing the SQL keywords recognized by the server. The **word** column contains the keyword

difference(text, text) returns int

- The **soundex()** function converts a string to its Soundex code. The **difference** function converts two strings to their Soundex codes and then reports the number of matching code positions. Since Soundex codes have four characters, the result ranges from zero to four, with zero being no match and four being an exact match.

```
=> SELECT word FROM pg_get_keywords() WHERE difference('dreams', word) >= 2;
    word
------------------
 columns
 current
 current_catalog
 current_date
 current_role
 current_schema
 current_time
 current_timestamp
 current_user
 database
 decimal
 definer
 delimiter
 delimiters
 depends
 …
```

Boulevard of broken **[plans | trims | routines | drops | froms | grants | groups]** :)

…

```
SELECT word, metaphone('dreams', 10), metaphone(word, 10),
levenshtein_less_equal(metaphone('dreams', 10), metaphone(word, 10), 3)
FROM pg_get_keywords()
ORDER BY 4 ASC LIMIT 40;

   word    | metaphone | metaphone | levenshtein_less_equal
-----------+-----------+-----------+------------------------
 trim      | TRMS      | TRM       |                      1
 from      | TRMS      | FRM       |                      2
 force     | TRMS      | FRS       |                      2
 time      | TRMS      | TM        |                      2
 normalize | TRMS      | NRMLS     |                      2
 freeze    | TRMS      | FRS       |                      2
 names     | TRMS      | NMS       |                      2
 schemas   | TRMS      | SKMS      |                      2
 temp      | TRMS      | TMP       |                      2
 trigger   | TRMS      | TRKR      |                      2
 types     | TRMS      | TPS       |                      2
 primary   | TRMS      | PRMR      |                      2
 drop      | TRMS      | TRP       |
…
```

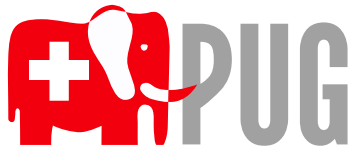
```
SELECT word, levenshtein_less_equal('dream', word, 3)
FROM pg_get_keywords()
ORDER BY 2 ASC LIMIT 40;

    word    | levenshtein_less_equal
------------+------------------------
 read       |                      2
 real       |                      2
 treat      |                      2
 desc       |                      3
 dec        |                      3
 from       |                      3
 data       |                      3
 trim       |                      3
 rename     |                      3
 array      |                      3
 drop       |                      3
 year       |                      3
 create     |                      3
 ref        |                      3
 day        |                      3
…
```

"O du lieber Augustin"

- O du lieber **Extension**, **Execute**, **Aggregate**,
- O du lieber **Committed**, alles ist hin.

"Du hast"

```
=> DO
$$
    DO CAST KEY!
    DO CAST KEY
       CASCADE!
    DO CAST KEY
       CASCADE
    AND EACH BY NULL
       DISCARD!
$$
```



Can PostgreSQL have a more prominent role in AI boom?

Josef Machytka

Can PostgreSQL have a more prominent role in the AI boom?

"Maybe in 20 years, everything will be done in PostgreSQL?"

(Simon Riggs, talk on PG conf EU 2023)

Pictures generated with DeamDreamGenerator

- Josef Machytka - NetApp

- 2024-06-27 Swiss PG day

PostgreSQL already has vector storage and vector search

- Extension pgvector, with vector data type, indexing methods and vector search.
- We have seen very nice talks about it here. I really enjoyed them!
- But every database is now advertising vector search…
- Can we do even better in PostgreSQL?

All the magic is happening inside an embedding model

- Embedding models are algorithms trained to encapsulate information into dense representations in a multi-dimensional space.

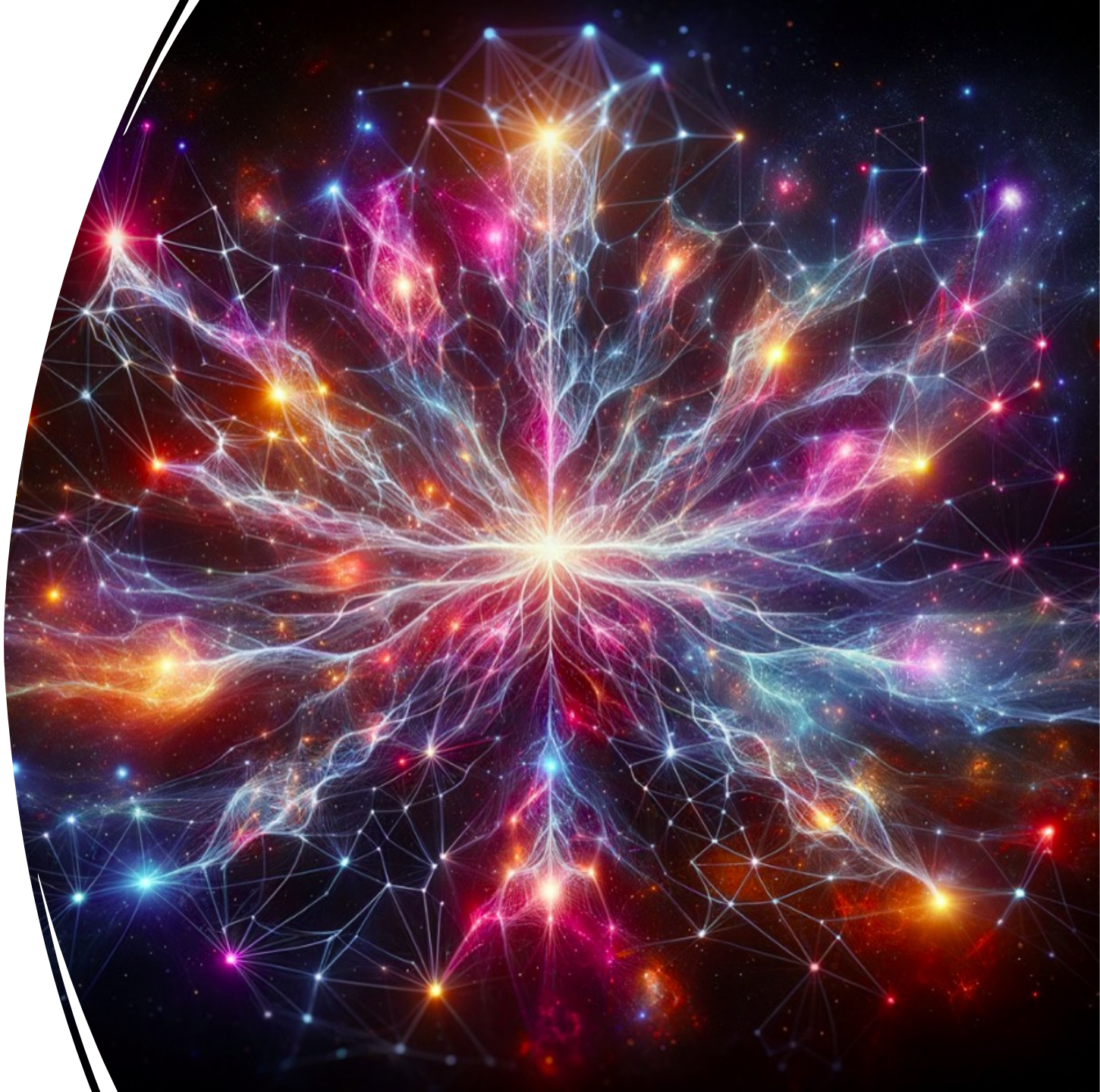
  (aws.amazon.com/what-is/embeddings-in-machine-learning)

- Vector embeddings are a way to convert words and sentences and other data into numbers that capture their meaning and relationships.

  (www.elastic.co/what-is/vector-embedding)

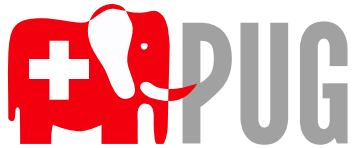
Neural networks and ML models are all about vectors and numbers

- Whole AI model is just a very advanced statistics.
- Stores data in its own binary format.
- TensorFlow ML model:
  - meta data of the graph structure
  - variables in checkpoint files (saved steps during training of the model)
  - index file
- They internally simulate graph database; they could easily hit different limits and performance problems.
- From big data perspective it is just another bunch of vectors. So maybe with PostgreSQL they can do better?



Databases obtain new functionality for ML, so why not AI?

- BigQuery already implemented ML and AI related functionality - you can use pretrained models directly in SQL
- PostgresML implements ML functionality
- New extensions for ML and AI models?
- Much more dimensions would be necessary – AI uses internally even billions of dimensions…
- I am not the ML expert; I am just asking…
- But maybe future really is more about PostgreSQL being everywhere…
- Could be also a very good new impulse for PostgreSQL improvements…



Visualizing Postgres buffers

Dickson Guedes

PG Day

PUG

https://github.com/guedes/pgviz_experiments/tree/files