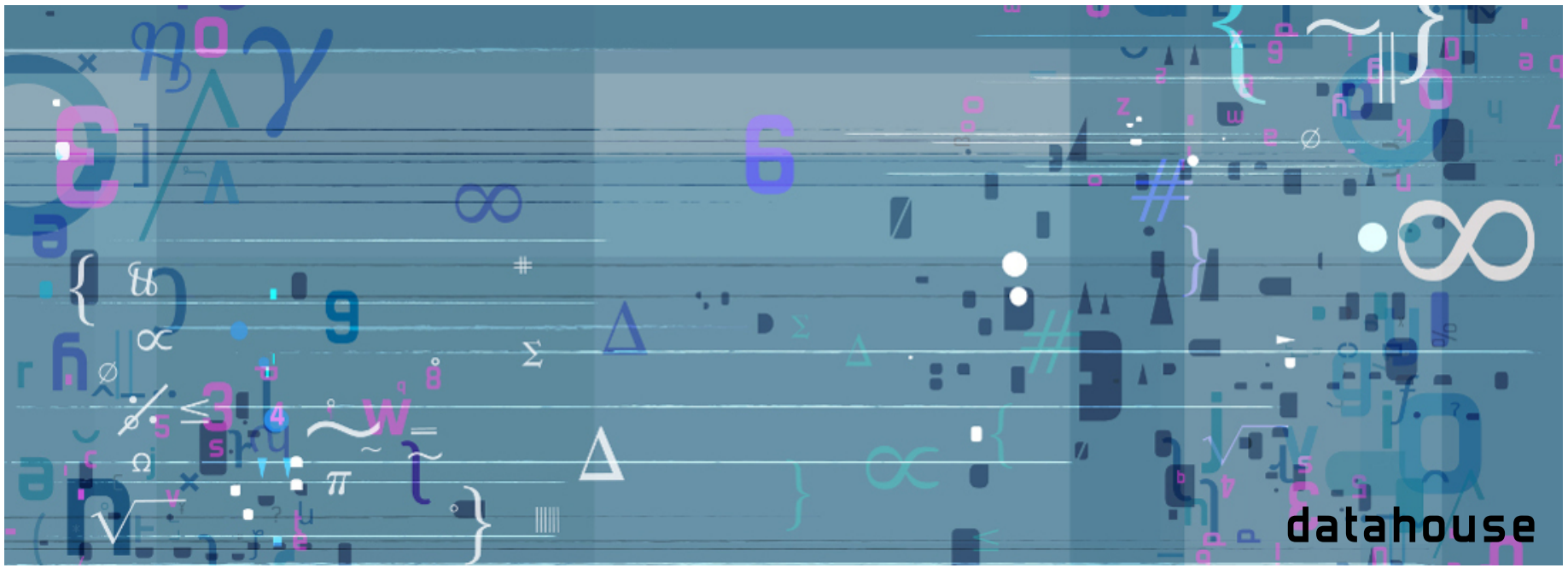
Von Business Logik und Datenbanken

Markus Wanner

Rapperswil , 30. Juni 2017



Einführung - Inhalt

Einführung

- Über den Autor
- Begriffe: Business Logik
- Begriffe: Datenbanken
- Begriffe: in

Allgemeiner Vergleich

- harte Fakten
- weiche Faktoren
- Tooling

Postgres Spezifika

- Procedural Languages
- Unit Testing
- Debugging
- Profiling

Fazit

- Entscheidungshilfen

Einführung - Über den Autor

Markus Wanner

- Senior Software Engineer, Datahouse AG
- Postgres-R
- Debian Developer (e.g. postgis)
- Präsident SwissPUG

Motivation

- hitzige Diskussionen, festgefahrene Meinungen (“Vietnam of Computer Engineering”)
- kaum hilfreiche Richtlinien

Einführung - Begriffe: Business Logik - Definition

Wikipedia

"In computer software, business logic [...] is the part of the program that encodes the real-world business rules that determine how data can be created, stored, and changed."

"It is contrasted with [...] lower-lever details of managing a database [...]"

business logic - prozedural

Nachdem der Besucher eingetroffen ist, begrüsse ich ihn und offeriere ihm einen Kaffee.

business rules - declarative

jeder Besucher muss höflich begrüsst werden

und: Kaffee darf auch Besuchern angeboten werden

Einführung - Begriffe: Business Logik - Unterscheidung

multi tier (PAC, MVC)

- data tier (model, abstraction)
- logic tier (controller)
- presentation tier (view)

point of view - what's your business?

Als Entwickler eines Datenbank-Systems, sind:

- Relationen und Attribute dann Business Objekte?
- ein Index Scan also Business Logik?
- Datenbanksysteme also das Business?

Einführung - Begriffe: Datenbank

Präzisierung

- “Database System” nicht bloss die Daten
- relational
- SQL - Martin Fowler: “Fundamentally I believe that SQL is the primary reason why relational databases have succeeded to the extent they have.”

think:

- PostgreSQL
- MySQL
- SQLite

Einführung - Begriffe: technische Sicht: “in”

Client-Server-Architektur

- separate Prozesse (OS)
- möglicherweise auf unterschiedlichen Hosts

Embedded - SQLite

- gleicher Prozess
- Datenbank als Library *in die Applikation gelinkt*
- prominente Beispiele: Chrome & Firefox, Android, Dropbox, Skype, etc...

Allgemeiner Vergleich - harte Fakten

Logik in der Datenbank

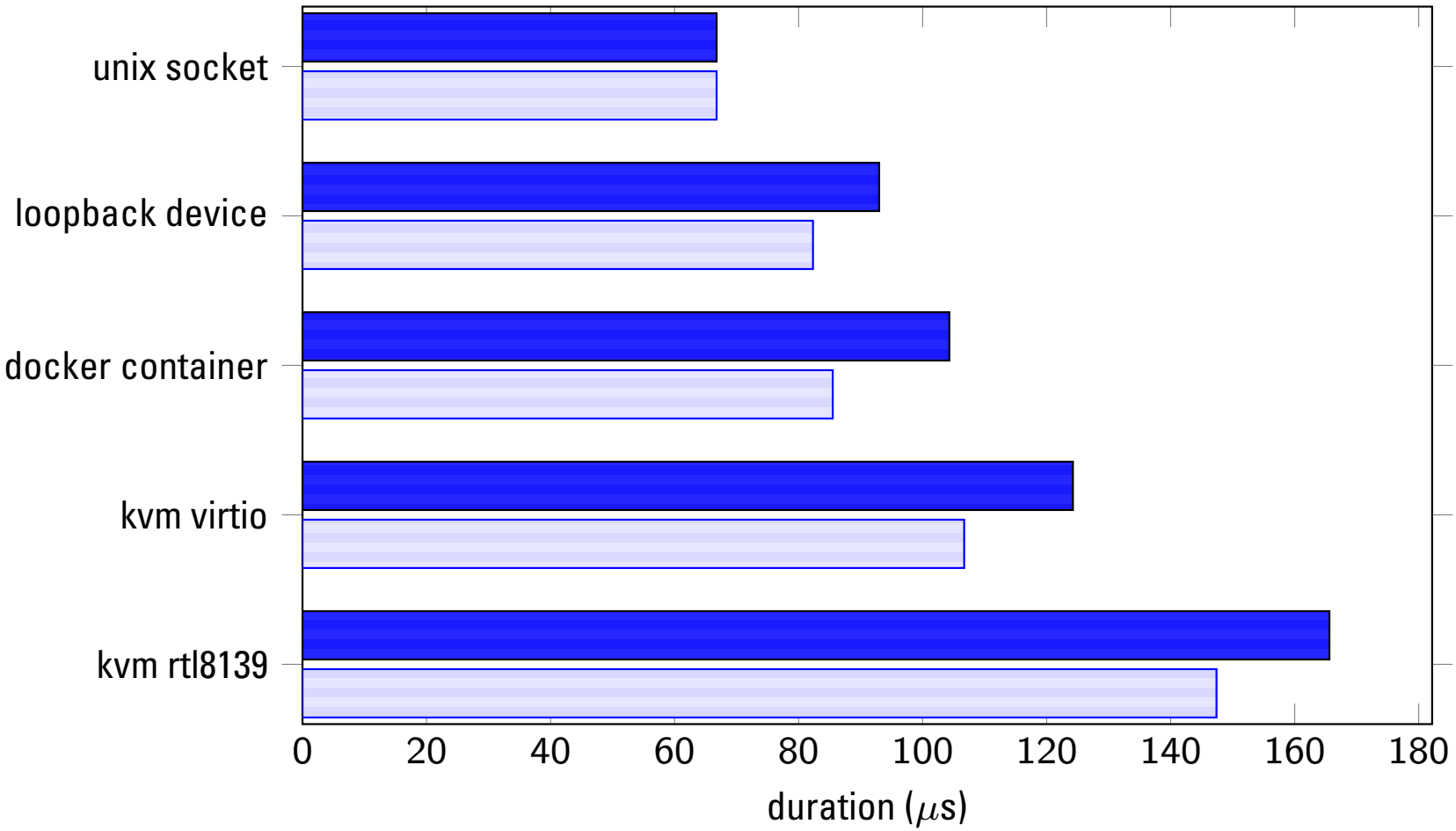
- nah an den Daten
- transaktional
- integriert in SQL
- limitiert auf SQL (Interface, als Sprache)
- vendor lock-in

Logik in der Applikation

- Freiheit: Programmiersprache
- Freiheit: Architektur (Multi-Threading, Error handling)
- Skalierbarkeit

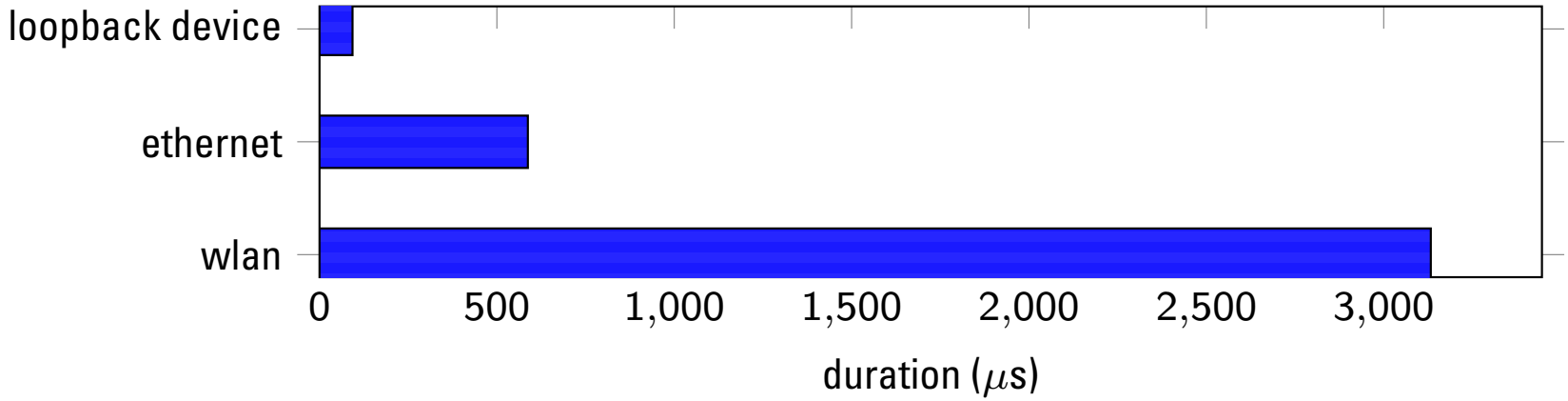
Allgemeiner Vergleich - harte Fakten - RTT

Roundtrip Times



Allgemeiner Vergleich - harte Fakten - RTT

Roundtrip Times



Messung

- gemittelt über 100000 einfache, sequenzielle SQL Abfragen
- eine einzelne TCP Verbindung

Allgemeiner Vergleich - weiche Faktoren

Team

- Kenntnisse und Vorlieben im Team
- potentiell verfügbare Entwickler

Konzepte

- deklarativ und funktional vs prozedural
- OR-Mapper
- CQRS
- REST (PostgREST)

Tooling

- IDE
- Versionskontrolle
- Performance: Profiling
- Debugging

Postgres Spezifika - Prozedurales Denken

```
CREATE FUNCTION ...()
...
AS $$
DECLARE
  result INT[];
BEGIN
  LOOP
    INSERT INTO ..
      RETURNING id INTO result[];
    i := i + 1;
  END

  RETURN result;
END
$$;
```

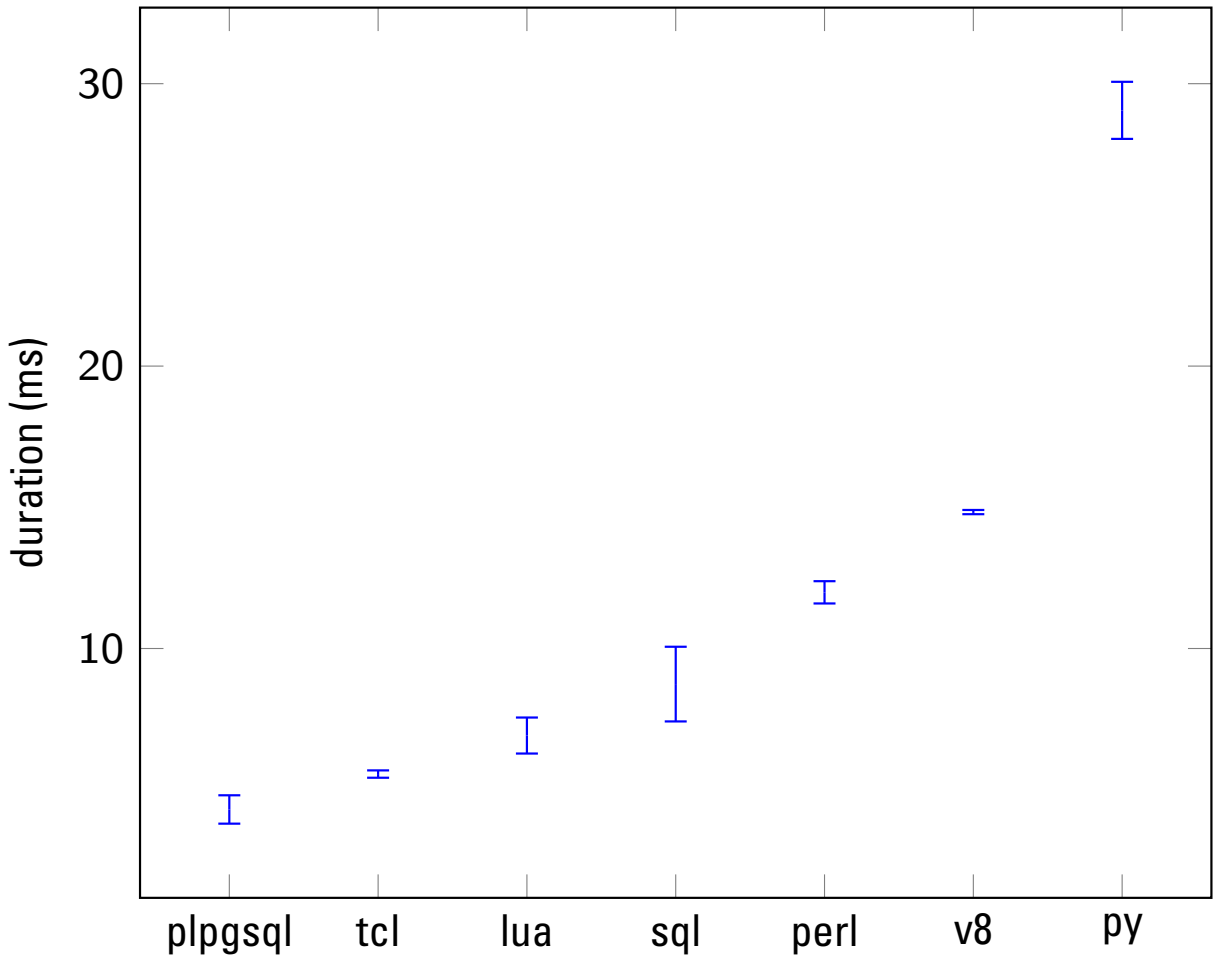
Postgres Spezifika - Procedural Languages

Eigenschaften

- via Extensions in div. Sprachen
- Unterschiedliche Qualität
- Context: Transaktion
- single Process, kein Multi-Threading
- Fehlerbehandlung via longjmp (panic)

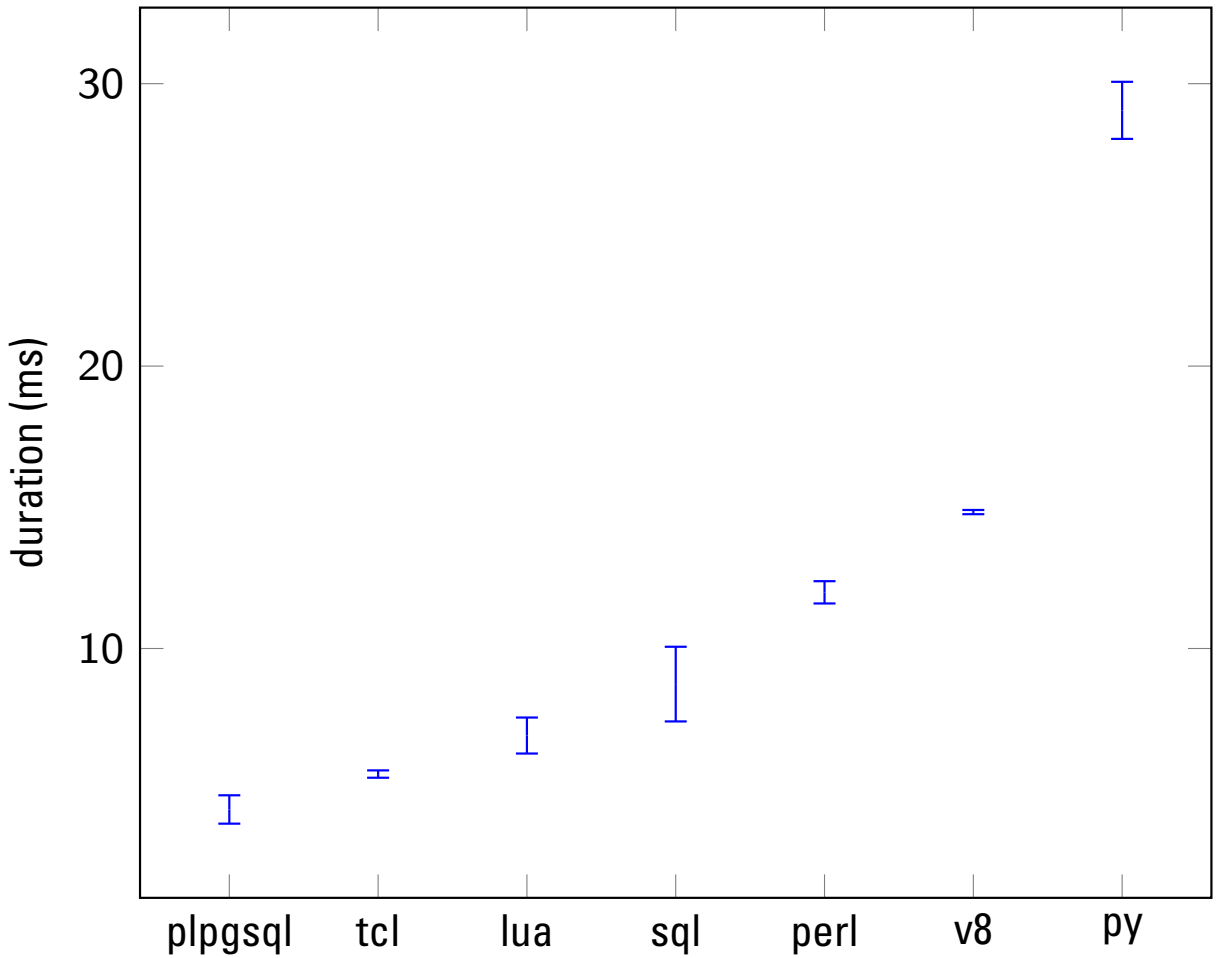
Business Logik in der Datenbank

Benchmarks - PL Startup Time



Business Logik in der Datenbank

Benchmarks - PL Startup Time



Ausreisser PL/R

196 ms (!)

Postgres Spezifika - Unit Testing

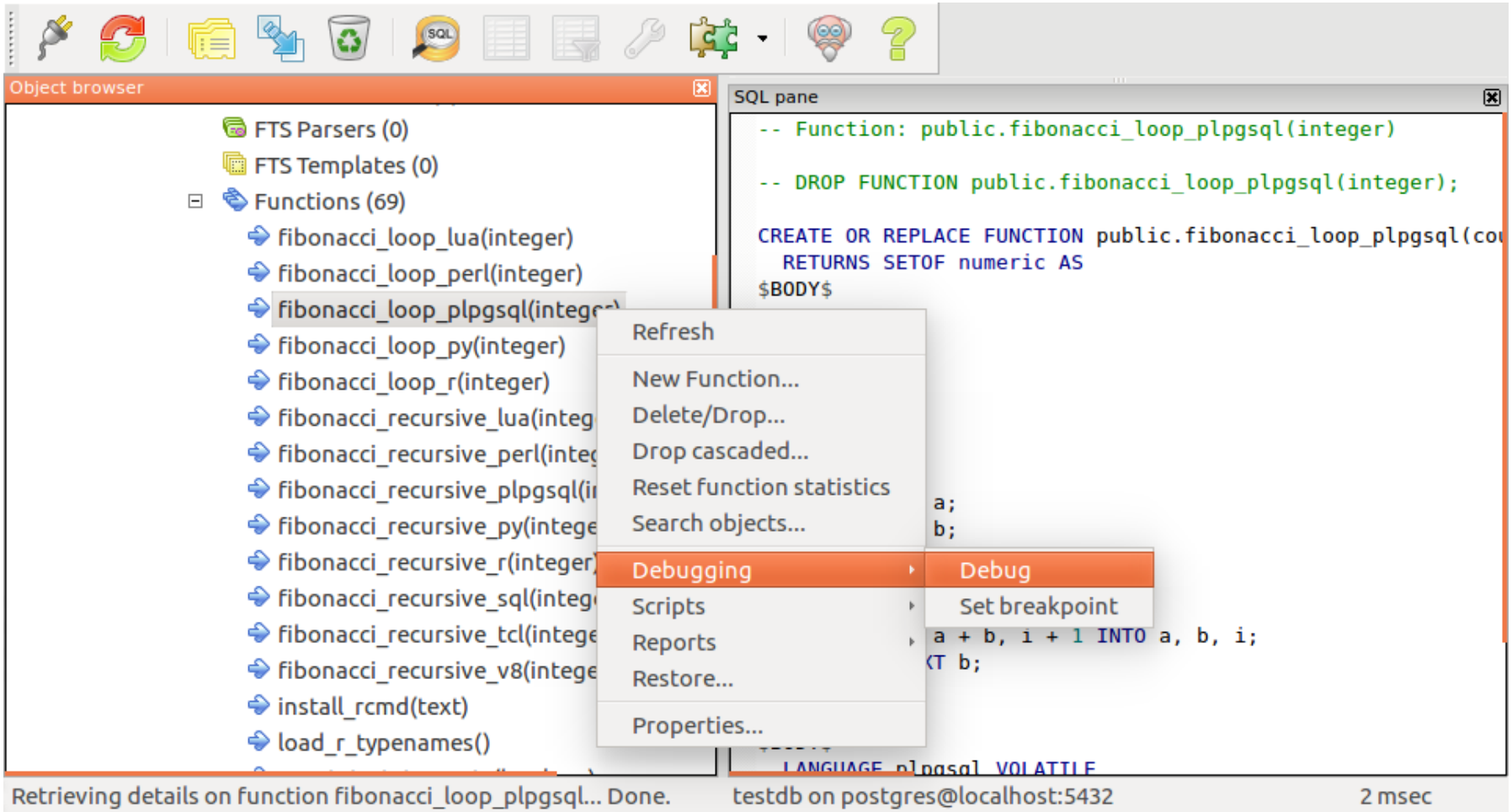
```
CREATE FUNCTION test_store_product()
RETURNS SETOF TEXT
AS $$
DECLARE
  setup RECORD;
BEGIN
  SELECT * FROM common_test_setup() INTO setup;

  PERFORM store_product(
    setup.ts_instance_1b_id,
    'MY_GRAND_PRODUCT',
    'MY_PRODUCT_TREE_NAME',
    '[
      {"tree_type": "str", "weight": 0.8},
      {"tree_type": "str", "weight": 0.2}
    ]'::JSON,
    '{}'::JSON
  );

  RETURN NEXT is(
    (SELECT array_agg(name) FROM
      (
        SELECT name
          FROM products
          WHERE ts_instance_id = setup.ts_instance_1b_id
          ORDER BY name
      ) AS x
    ),
    ARRAY['ex_prod_chf_01', 'ex_prod_chf_03', 'MY_GRAND_PRODUCT']
  );
END
$$ LANGUAGE plpgsql;
```

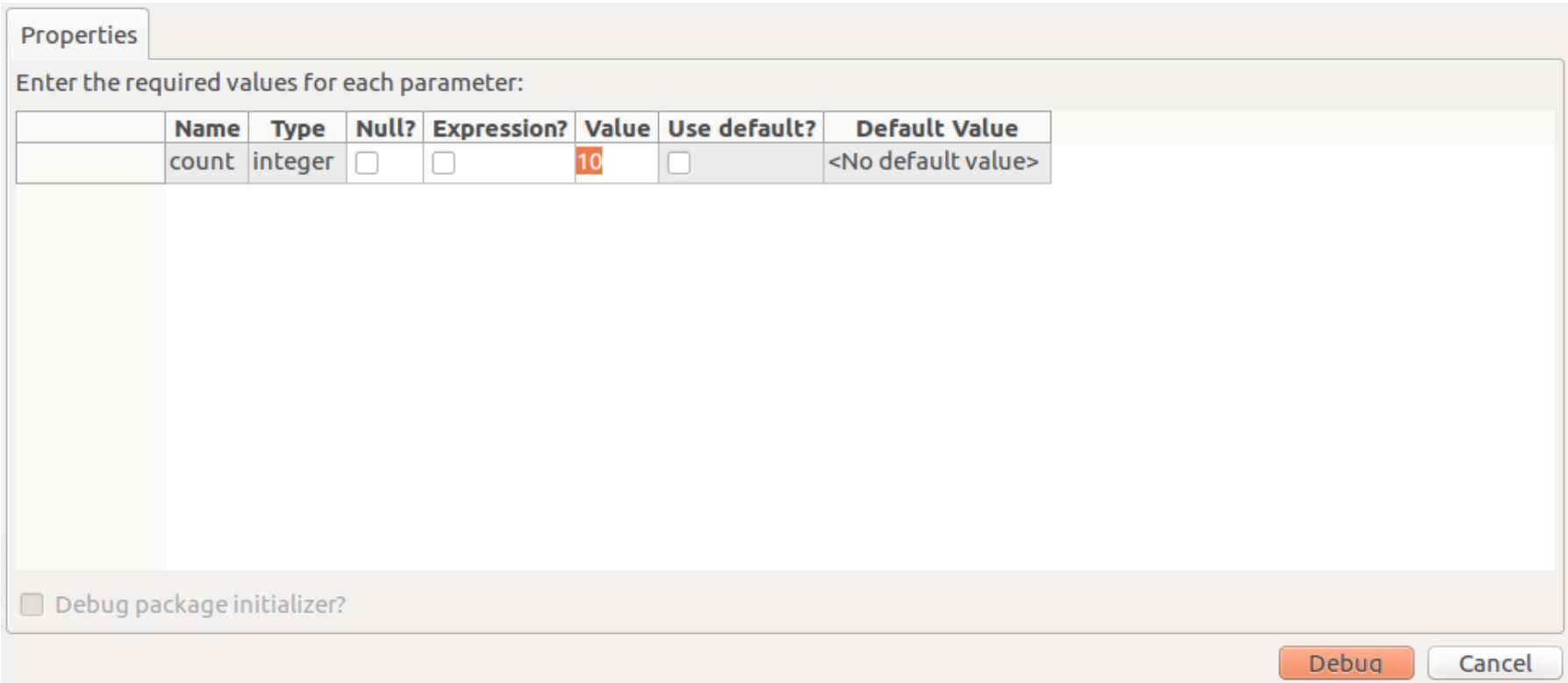
Postgres Spezifika - Debugging

plprofiler Beispiel: get_utm_zone



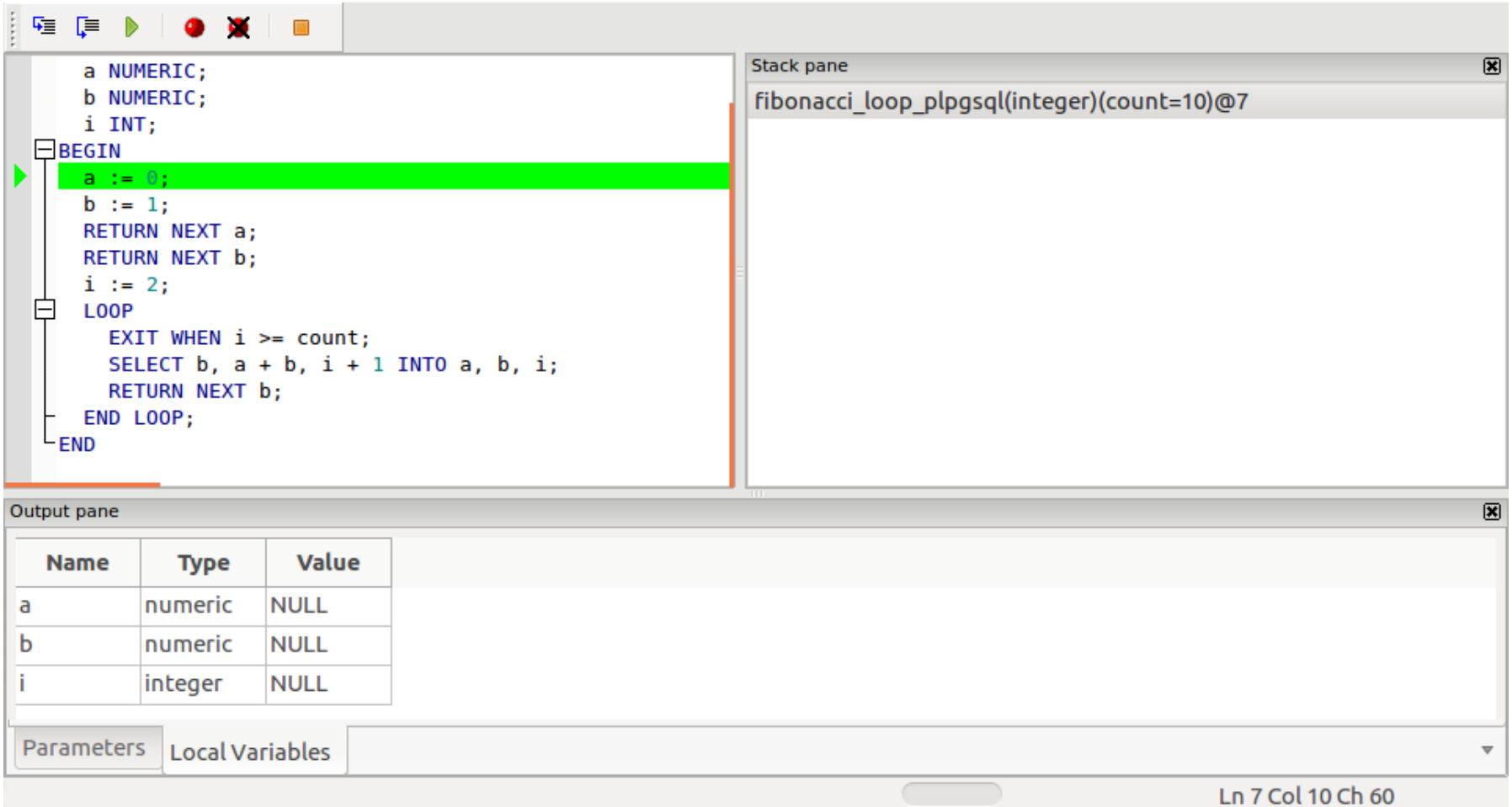
Postgres Spezifika - Debugging

plprofiler Beispiel: get_utm_zone



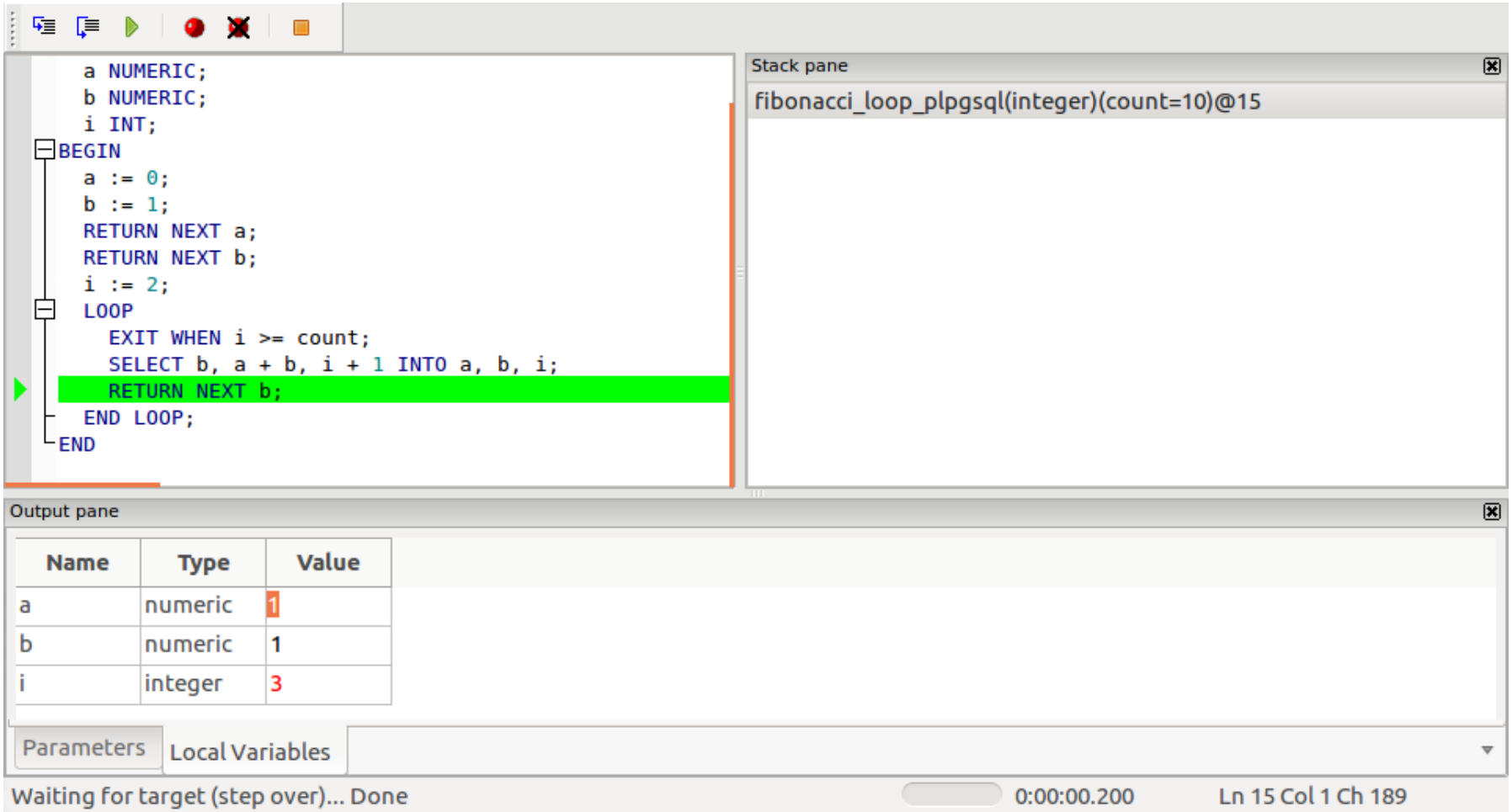
Postgres Spezifika - Debugging

plprofiler Beispiel: get_utm_zone



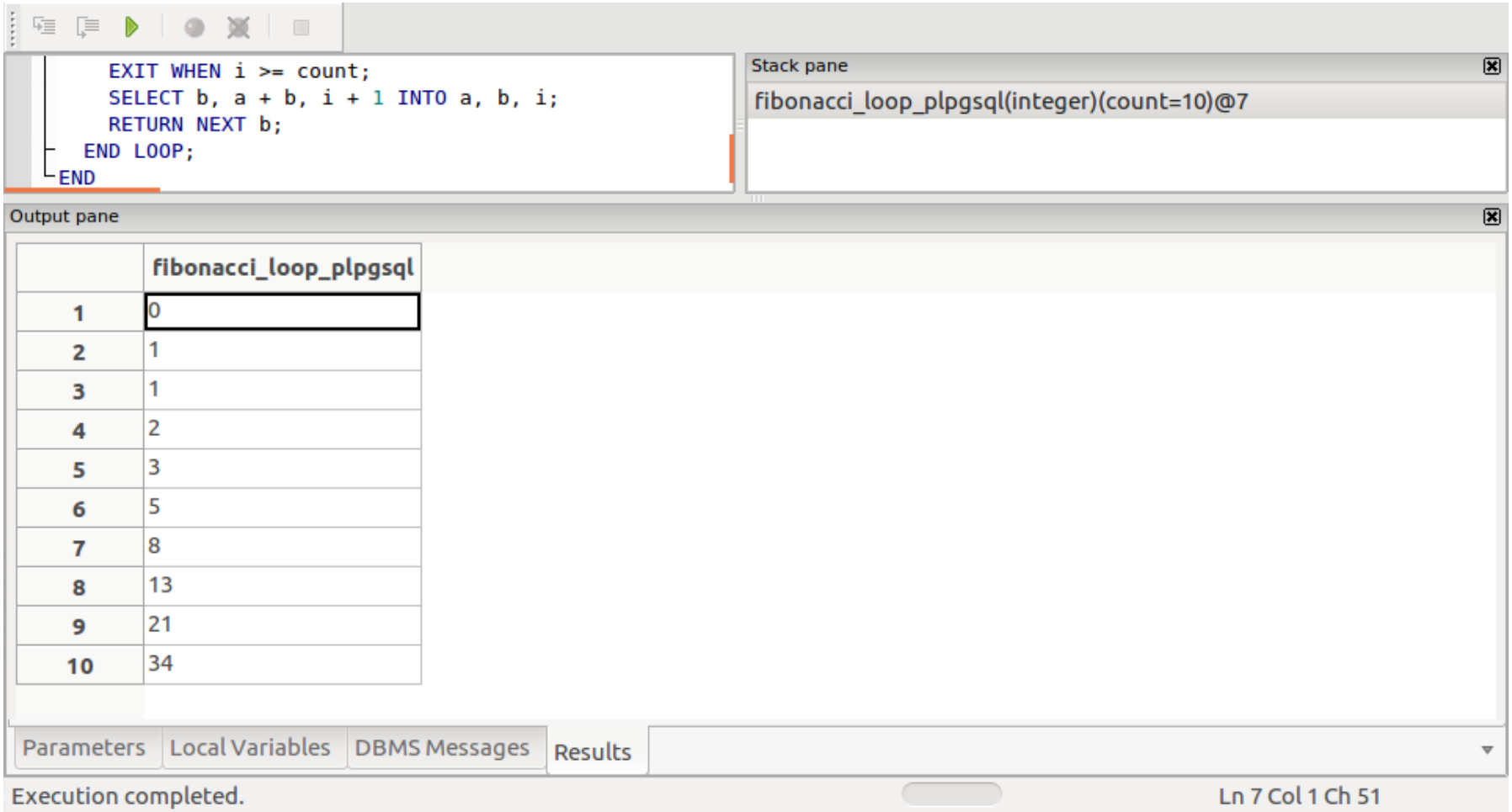
Postgres Spezifika - Debugging

plprofiler Beispiel: get_utm_zone



Postgres Spezifika - Debugging

plprofiler Beispiel: get_utm_zone



Postgres Spezifika - Profiling

plprofiler Beispiel: get_utm_zone

Function public.get_utm_zone() oid=4077674 (hide)

self_time = 17,322 µs

total_time = 17,322 µs

```
public.get_utm_zone (in_geom thirdparty.geometry)
    RETURNS text
```

Line	exec_count	total_time	longest_time	Source Code
0	629	17,322 µs (100.00%)	709 µs	-- Function Totals
1	0	0 µs (0.00%)	0 µs	
2	0	0 µs (0.00%)	0 µs	DECLARE
3	0	0 µs (0.00%)	0 µs	geomgeog geometry;
4	0	0 µs (0.00%)	0 µs	zone INT;
5	0	0 µs (0.00%)	0 µs	latband INT;
6	0	0 µs (0.00%)	0 µs	BEGIN
7	629	3,092 µs (17.85%)	110 µs	geomgeog := ST_Transform(in_geom, 4326);
8	0	0 µs (0.00%)	0 µs	
9	629	3,876 µs (22.38%)	240 µs	zone := (floor((ST_X(geomgeog) + 180) / 6) + 1)::INT;
10	629	2,425 µs (14.00%)	81 µs	latband := floor((ST_Y(geomgeog) + 80) / 8)::INT;
11	0	0 µs (0.00%)	0 µs	
12	0	0 µs (0.00%)	0 µs	-- 'I' and 'O' are omitted
13	629	2,150 µs (12.41%)	119 µs	IF latband >= 6 THEN
14	629	946 µs (5.46%)	48 µs	latband := latband + 1;
15	0	0 µs (0.00%)	0 µs	END IF;
16	0	0 µs (0.00%)	0 µs	
17	629	1,649 µs (9.52%)	83 µs	IF latband >= 12 THEN
18	629	832 µs (4.80%)	77 µs	latband := latband + 1;
19	0	0 µs (0.00%)	0 µs	END IF;
20	0	0 µs (0.00%)	0 µs	
21	629	2,787 µs (16.09%)	125 µs	RETURN zone::TEXT || chr(ascii('C') + latband);
22	0	0 µs (0.00%)	0 µs	END;
23	0	0 µs (0.00%)	0 µs	

Fazit

pro DB

- Daten intensiv (und aggregierend)?
- deklarative oder funktionale Beschreibung
- Mehrwert durch Nutzung aus SQL

pro App

- CPU-intensiv (trotz geringen Datenmengen)
- Verarbeitung von Daten ausserhalb der Db
- User Interface spezifisch

Herzlichen Dank für Ihre Aufmerksamkeit.

Bei Fragen stehe ich Ihnen gerne zur Verfügung.

Markus Wanner
+41 44 289 92 42
markus.wanner@datahouse.ch
www.datahouse.ch

Datahouse AG
Alte Börse
Bleicherweg 5
CH-8001 Zürich