THINKWISE

PostgreSQL: Is dit jouw volgende database?

Mark Jongeling
Software Architect

Davy van der Schoot
Software Engineer

Agenda

- **De olifant in de kamer**

  Waarom PostgreSQL en waarom nu?

- **PL/pgSQL vs T-SQL**

  Verschillen en overeenkomsten

- **Concurrency in PostgreSQL**

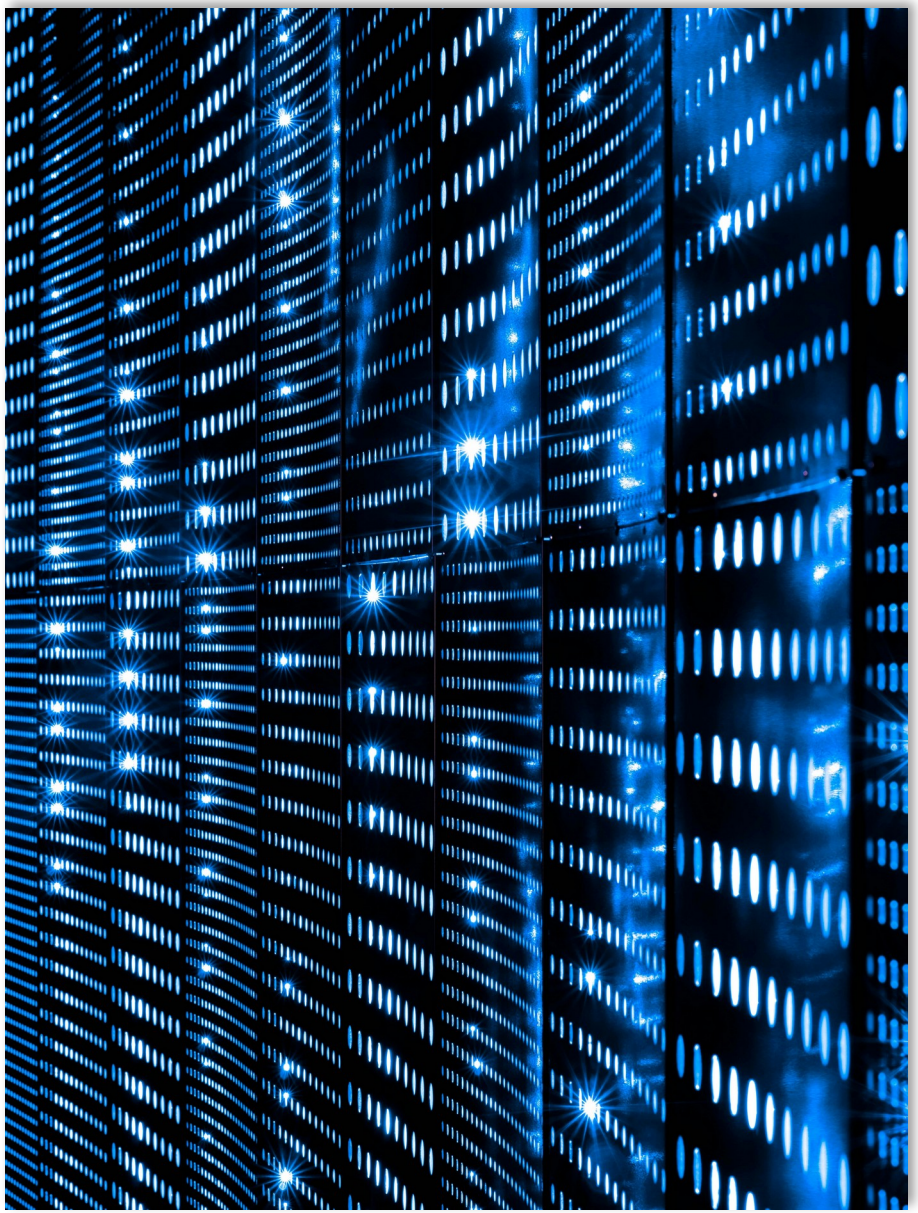
  Multi-versioning en isolation levels

- **Extension landscape**

  Plug-and-play nieuwe functionaliteiten in een handomdraai

- **Is dit jouw volgende database?**

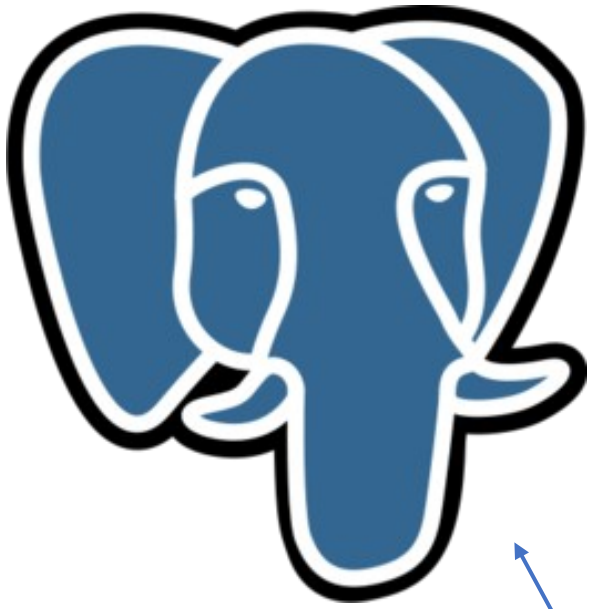
  Vooruitkijken naar de toekomst



Waarom PostgreSQL en waarom nu?

De olifant in de kamer

- PostgreSQL (of Postgres)
- Relationele database
- Open source
- Géén licentiekosten
- Cloud-native
- Geen vendor lock-in



Slonik

PL/pgSQL vs T-SQL

PL/pgSQL vs T-SQL:
Is alles anders?

SQL Server

```
select c.customer_name, count(*) as order_count
from sales_order o
join customer c
  on c.customer_id = o.customer_id
group by c.customer_name
order by order_count desc;
```

PostgreSQL

```
select c.customer_name, count(*) as order_count
from sales_order o
join customer c
  on c.customer_id = o.customer_id
group by c.customer_name
order by order_count desc;
```

Paginering

SQL Server

```
select c.customer_name
from customer c
order by customer_name
offset 0 rows fetch next 10 rows only;
```

PostgreSQL

```
select c.customer_name
from customer c
order by c.customer_name
limit 10 offset 0;
```

Aggregatie van tekst

SQL Server

```
select c.customer_name,
       string_agg(o.order_number, ', ') as orders
from customer c
join sales_order o
  on o.customer_id = c.customer_id
group by c.customer_name;
```

PostgreSQL

Upsert:
Toevoegen of updaten in één statement

SQL Server

```
merge into customer as target
using
(   select 123 as customer_id,
           'Etos' as customer_name
) as source
on target.customer_id = source.customer_id
when matched then
    update
    set customer_name = source.customer_name
when not matched then
    insert (customer_id,
            customer_name)
    values (source.customer_id,
            source.customer_name);
```

PostgreSQL

```
insert into customer (customer_id, customer_name)
values (123, 'Etos')
on conflict (customer_id)
do update set customer_name = excluded.customer_name;
```

JSON

SQL Server

```sql
select
    customer_id,
    json_value(profile_data, '$.email') as email
from customer
where json_value(profile_data, '$.country') = 'NL';
```

```
{
  "email": "devday@example.com",
  "country": "NL"
}
```

PostgreSQL

```sql
select customer_id,
       profile_data->>'email' as email
from customer
where profile_data->>'country' = 'NL';
```

```
devday@example.com
```

Operator	Output
->	JSON value
->>	Text value

Arrays

SQL Server

```
select *
from article
where exists (
    select 1
    from openjson(tags)
    where value = 'featured'
);
```

- JSON
- XML
- Platte tekst

PostgreSQL

```
select *
from article
where 'Featured' = any(tags);
```

- json[]
- boolean[]
- text[]
- varchar[]
- numeric[]
- integer[][] (tic-tac-toe)
- …
- en zelfs eigen types

Arrays

```
135  create temporary table article
136  (
137     article_name varchar(50),
138     tags varchar[]
139  );
140  insert into article
141  values ('Apple', array['Awesome', 'Featured', 'Healthy']),
142         ('Banana', array['Curved', 'Healthy']);
143
144  select *
145  from article
146  where 'Featured' = any(tags);
147
```

Data Output | Messages | Notifications

SQL

	article_name character varying (50)	tags character varying[]
1	Apple	{Awesome,Featured,Healthy}

Een paar andere verschillen

Onderwerp	T-SQL	PL/pgSQL
Temp data	#temp tables, table variables	CTE's, Record, Functions (setof), %rowtype , create temporary table (like table including all)
Error handling	try…catch	exception when
Dynamic SQL	sp_executesql	execute format
Return types	Beperkter	Flexibel (JSON, arrays, custom types)

Functions

```sql
create or replace function get_orders_for_customer(p_customer_id int)
returns setof orders
language plpgsql
as $$
begin
    return query
    select *
    from orders
    where customer_id = p_customer_id;
end;
$$;

select * from get_orders_for_customer(42);
```

```sql
create or replace function extract_emails(payload jsonb)
returns text[]
language sql
as $$
    select array_agg(value->>'email')
    from jsonb_array_elements(payload->'users');
$$;

select extract_emails('{
  "users": [
    {"email": "a@test.com"},
    {"email": "b@test.com"}
  ]
}'::jsonb);
```

```sql
create or replace function filter_expensive_orders(payload jsonb, min_amount numeric)
returns jsonb
language sql
as $$
    select jsonb_agg(o)
    from jsonb_array_elements(payload->'orders') as o
    where (o->>'amount')::numeric >= min_amount;
$$;
```

```sql
create or replace function count_rows(tablename text)
returns bigint
language plpgsql
as $$
declare
    result bigint;
begin
    execute format('select count(*) from %I', tablename)
    into result;

    return result;
end;
$$;

select count_rows('orders');
```

Triggers
SQL Server

When	Event	Row-level	Statement-level
BEFORE	INSERT/UPDATE/DELETE	—	—
	TRUNCATE	—	—
AFTER	INSERT/UPDATE/DELETE	—	Tables
	TRUNCATE	—	—
INSTEAD OF	INSERT/UPDATE/DELETE	—	Tables Views
	TRUNCATE	—	—

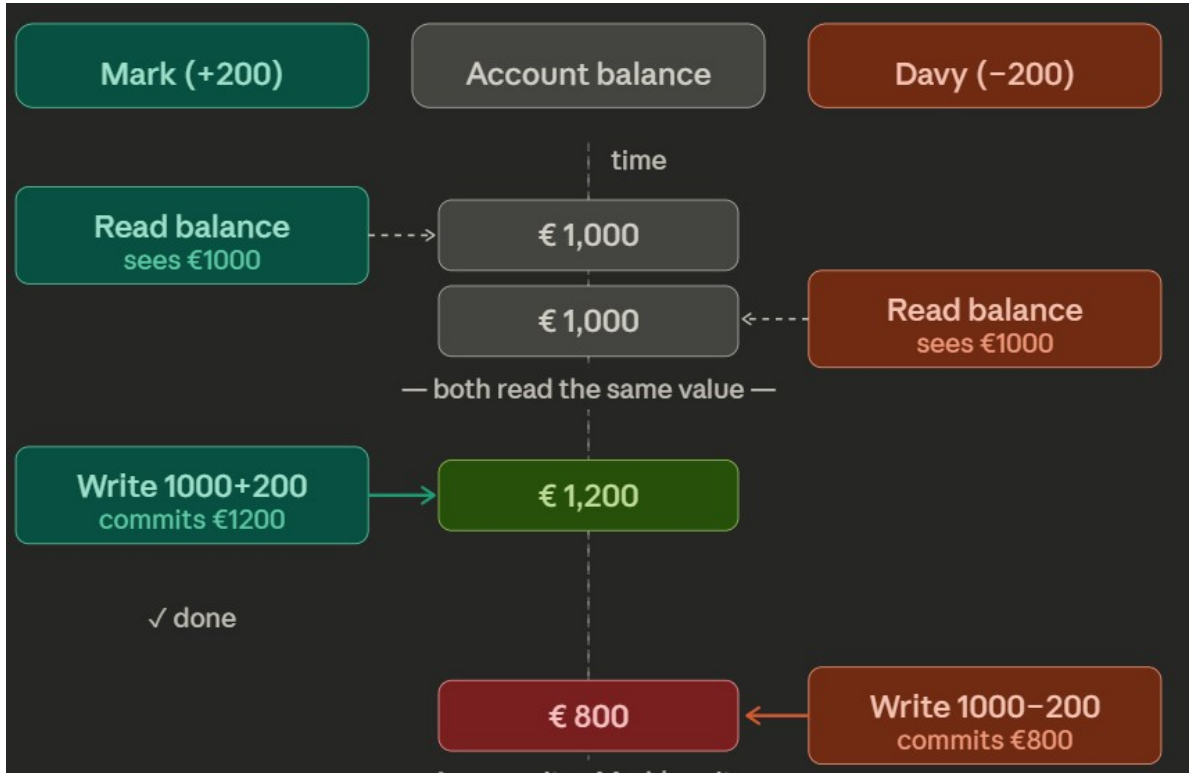
Triggers
PostgreSQL

When	Event	Row-level	Statement-level
BEFORE	INSERT/UPDATE/DELETE	Tables and foreign tables	Tables, views, and foreign tables
	TRUNCATE	—	Tables and foreign tables
AFTER	INSERT/UPDATE/DELETE	Tables and foreign tables	Tables, views, and foreign tables
	TRUNCATE	—	Tables and foreign tables
INSTEAD OF	INSERT/UPDATE/DELETE	Views	—
	TRUNCATE	—	—

Concurrency: Hoe gaat PostgreSQL om met reads en writes

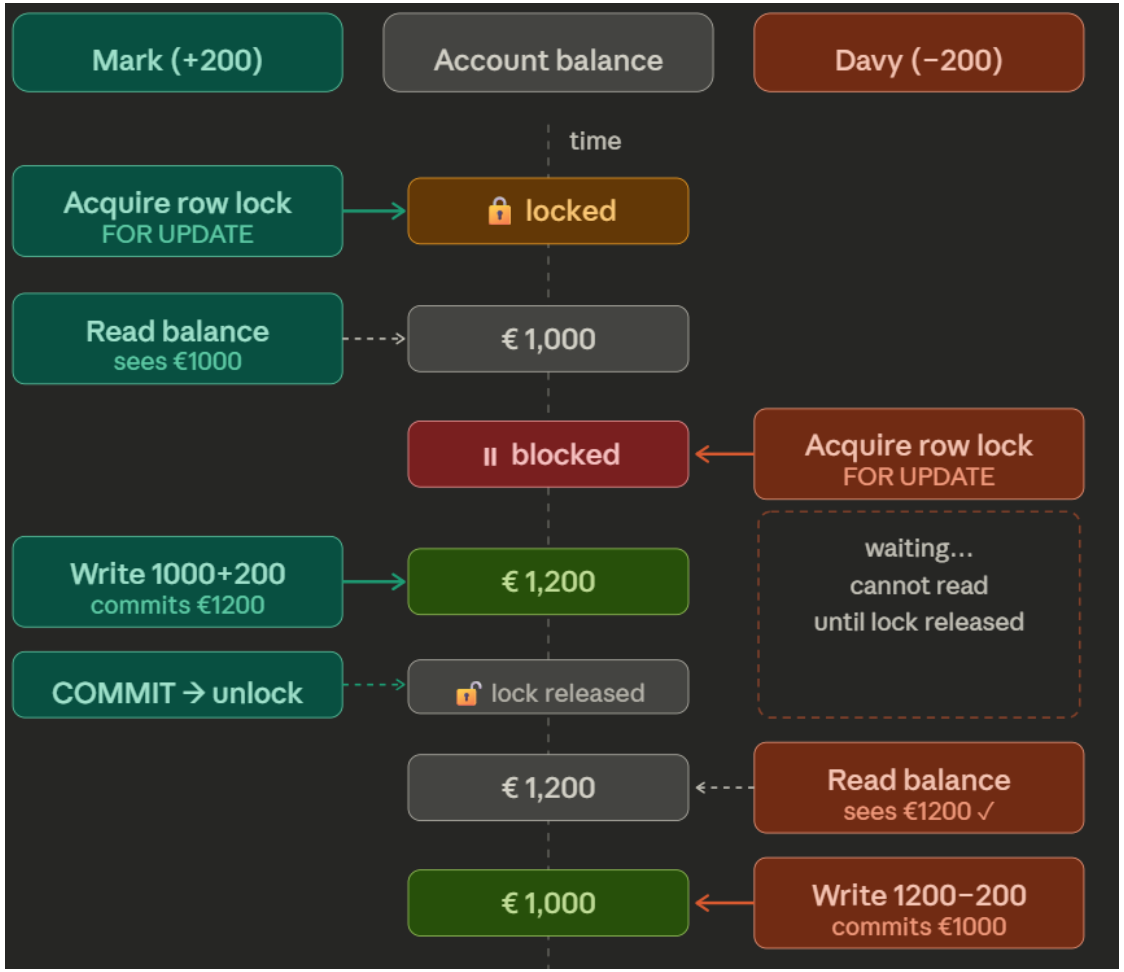
Concurrency Control: Wat is concurrency?

- Meerdere gebruikers
- Meerdere transacties tegelijk
- Transacties overschrijven elkaar
- Data loss



SQL Server: Pessimisme als standaard

- Exclusive locks blokkeren shared locks
- Stel ik wilde alleen lezen, dan moest ik ook wachten
- Gevolg: Performance degradatie en deadlocks



SQL Server: Opt-in Optimisme

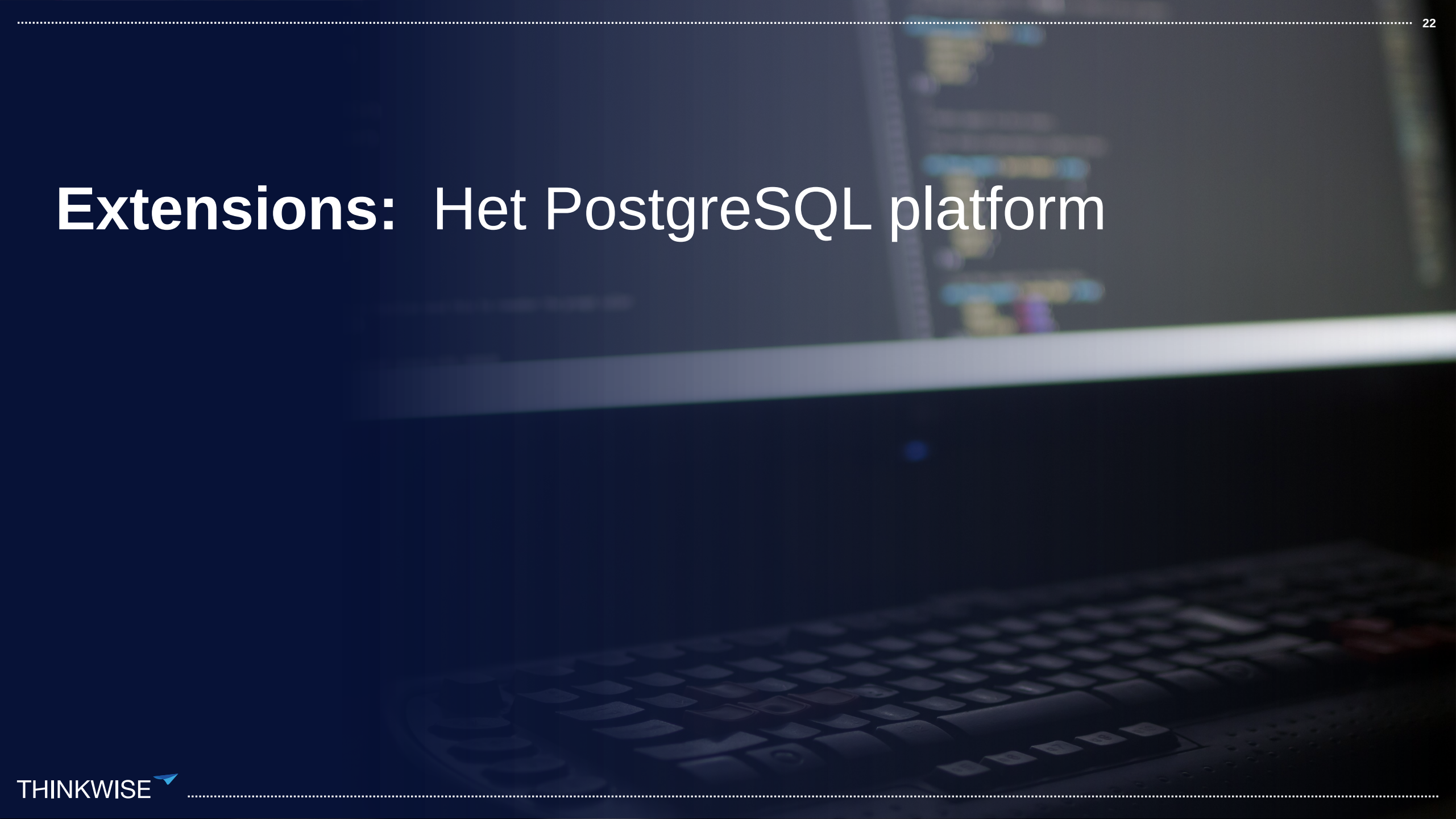
- Read Commited Snapshot Isolation, Snapshot Isolation
- Schrijf transacties : Kopie van het origineel in tempdb
- Lees transacties: Lees versie in tempdb
- Readers en writers uit elkaar getrokken
- Nadeel: tempdb I/O, extra configuratie

```
⚠ Transaction (id=2441927 xsn=753751 spid=85 elapsed_time=14052) has been marked as victim and
  it will be rolled back if it accesses the version store.
```

PostgreSQL: Multi-Version als de standaard

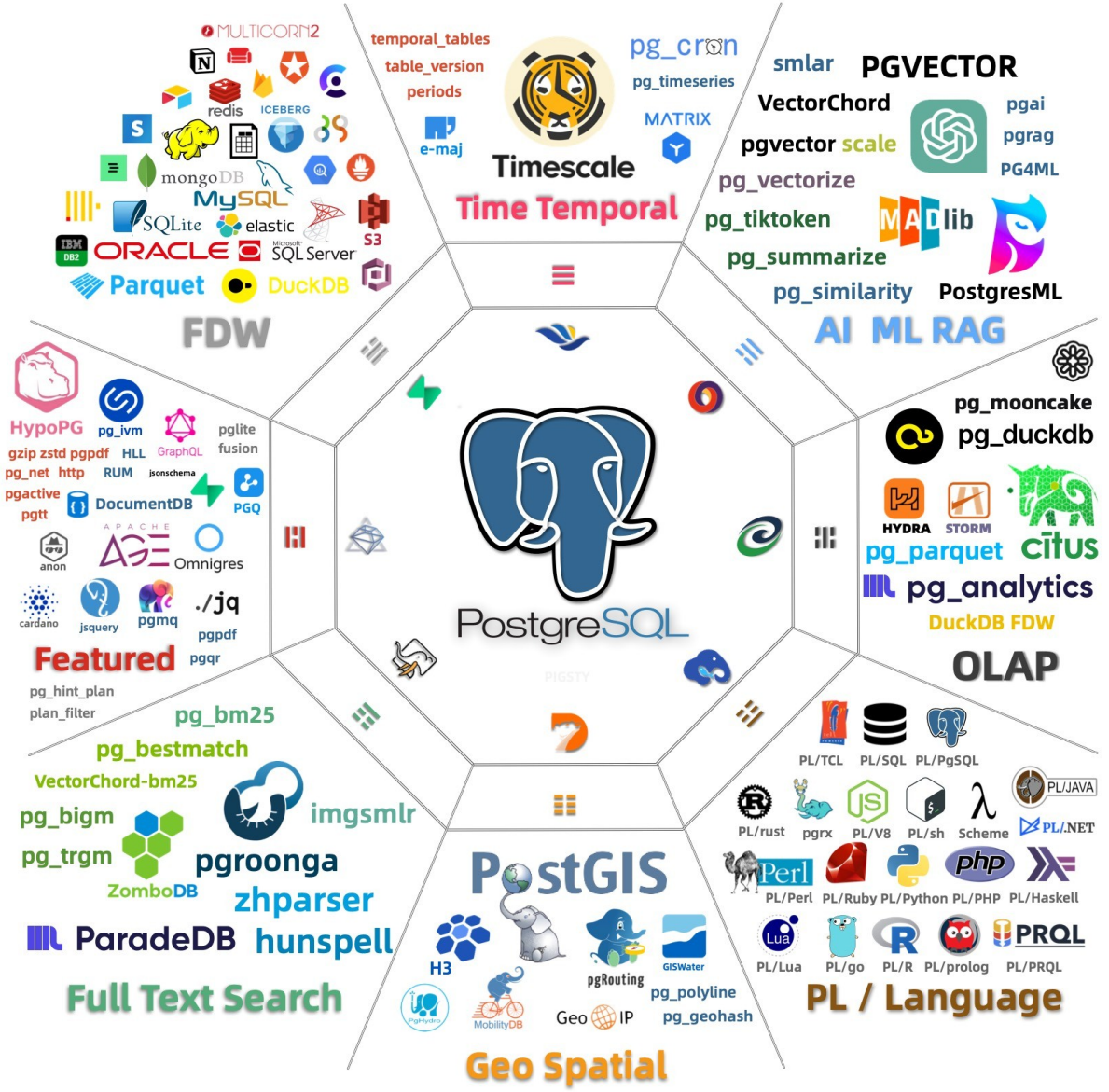
- Versies bestaan in de tabel
- Geen extra configuratie
- Geen tempdb bottleneck
- Minder deadlocks en minder verrassingen

Extensions: Het PostgreSQL platform



Extensions: Het PostgreSQL platform

- Enorm extensie ecosysteem
- Extensions zijn plug and play ingebouwde features
- Nieuwe indexes
- Nieuwe datatypes
- Nieuwe talen

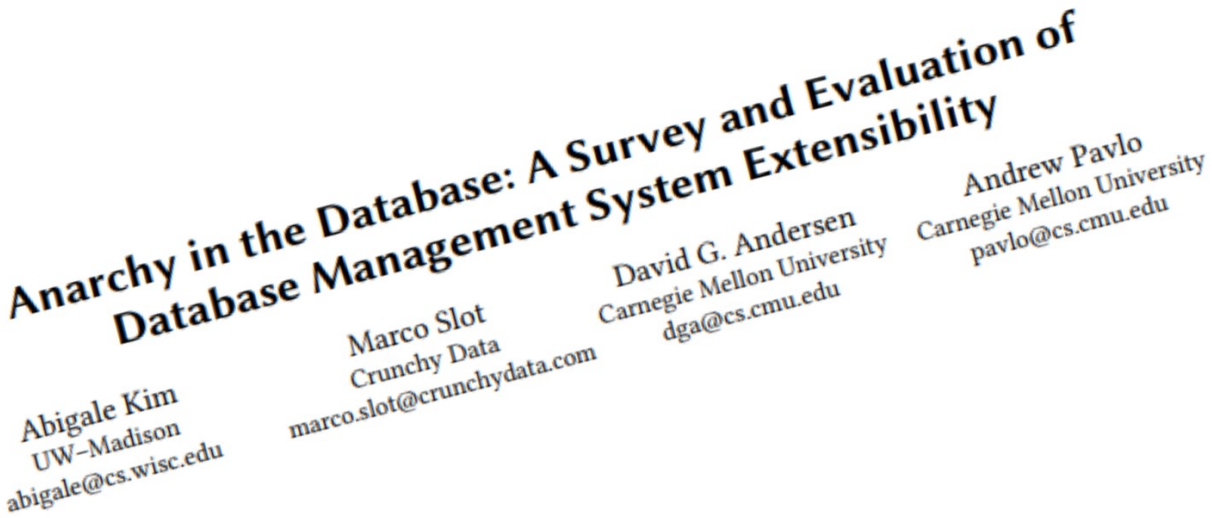


Extensions: De essentials

- PostGIS voor Geospatial data
- Pgvector voor vector data
- pg_stat_statements voor statistics tracking

Extensions: Security & Compatibility

- Compatibility niet gegarandeerd
- Cloud providers
  - Custom whitelists
  - Source code modification



Anarchy in the Database: A Survey and Evaluation of Database Management System Extensibility

Abigale Kim
UW–Madison
abigale@cs.wisc.edu

Marco Slot
Crunchy Data
marco.slot@crunchydata.com

David G. Andersen
Carnegie Mellon University
dga@cs.cmu.edu

Andrew Pavlo
Carnegie Mellon University
pavlo@cs.cmu.edu

Is dit jouw volgende database?

Is dit jouw volgende database?

- Kostenbesparing op licentie
- Toekomstbestendig door grote open source community
- Bewezen, krachtig, uitbreidbaar, cloud-native