Lifetime of a SELECT

A blick into the depth of PostgreSQL internals

Sergey Dudoladov
Zalando SE

zalando

Who am I ?

A Senior Database Engineer in Zalando

Earlier more infra, now more consultancy

Volunteer in the PostgreSQL community

zalando

Zalando SE

- In-house PostgreSQL as a Service
- > 3000 PostgreSQL clusters
- **Popular open source projects:**
  - Postgres-operator
  - Spilo

zalando

Who are you ?

Developers with some PostgreSQL experience who want to understand what happens under the hood

Why this talk ?

PostgreSQL internals are fun

Repetitive questions are not

Money saved on avoidable incidents is nice

Philosophy

KISS

More why than how

A problem is more important than a solution

zalando

All models are wrong, but some are useful.

George E. P. Box, a British statistician

The plan

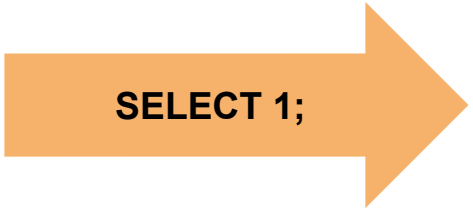
How PostgreSQL runs

How PostgreSQL runs SELECTs

How PostgreSQL runs SELECTs fast

SELECT 1 as hello_pg_meetup;

How PostgreSQL runs



No DB is 100% available

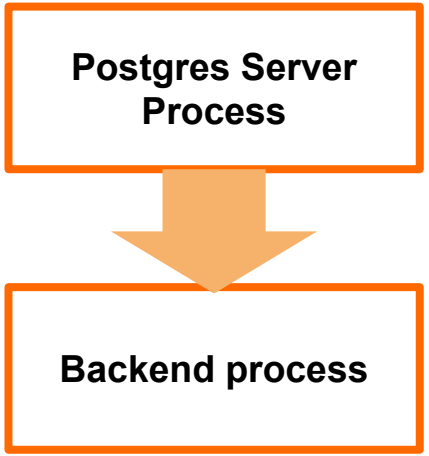
Why you will forget

PostgreSQL is very reliable

Infrastructure is rapidly maturing

“Hay in a haystack”

Process model



Connections
come with a cost

Typical issues

Opening and closing many short-lived connections. Fine if load is low.

Hitting max_connections

Connection not terminated

TODOs

Monitor!

Connection pooler is your ❤️ ❤️ ❤️

Teach yourself to cancel backends gracefully. Hint: pg_(cancel|terminate)_backend()

Have a way to nicely restart an application and a connection pool

zalando

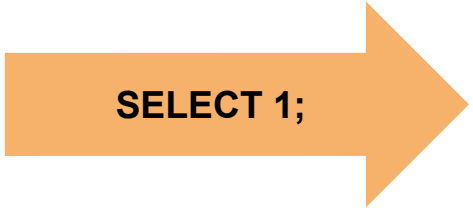
The Lifetime

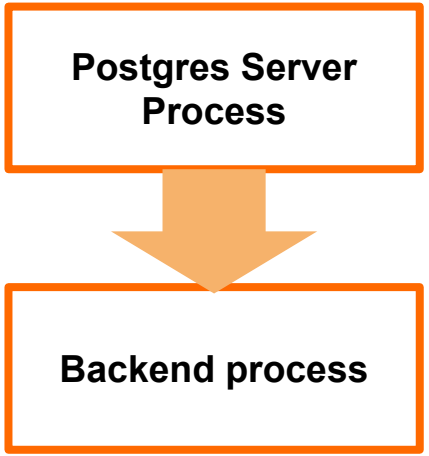
How PostgreSQL runs

```
EXPLAIN ANALYZE SELECT 1 AS hello_pg_meetup;
                                QUERY PLAN
═══════════════════════════════════════════════════════
 Result  (cost=0.00..0.01 rows=1 width=4) (actual
time=0.002..0.003 rows=1.00 loops=1)
 Planning Time: 0.027 ms
 Execution Time: 0.018 ms
(3 rows)
```

zalando

Mental model v1.0





The plan

How PostgreSQL runs

How PostgreSQL runs SELECTs

How PostgreSQL runs SELECTs fast

```
SELECT * FROM t;
```

… If you've chosen the right data structures and organized things well, the algorithms will almost always be self-evident …

Rob Pike’s
5 Rules of Programming

User’s table

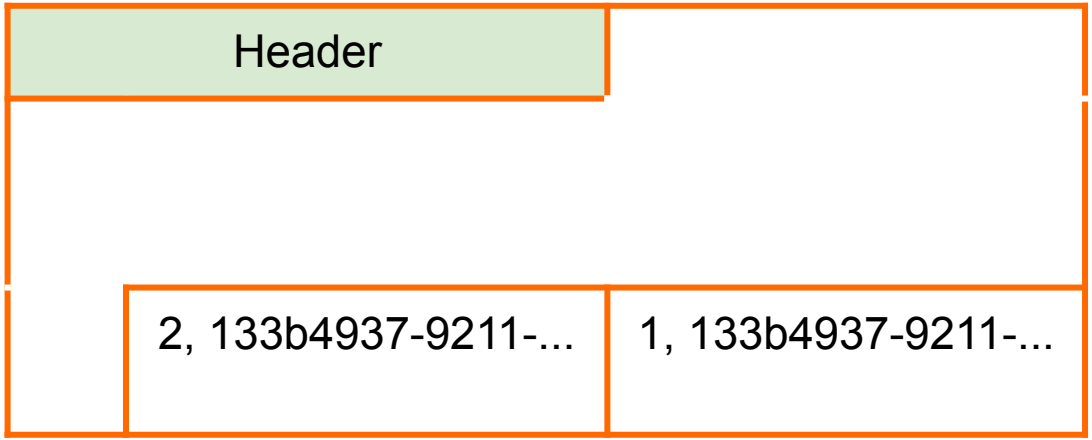
```
CREATE TABLE t (
        id SERIAL PRIMARY KEY,
        payload TEXT
);

INSERT INTO t (payload)
SELECT gen_random_uuid()::TEXT
FROM generate_series(1, 10000000);
```

* You need pgcrypto for gen_random_uuid()
** Do you see the problem here ?

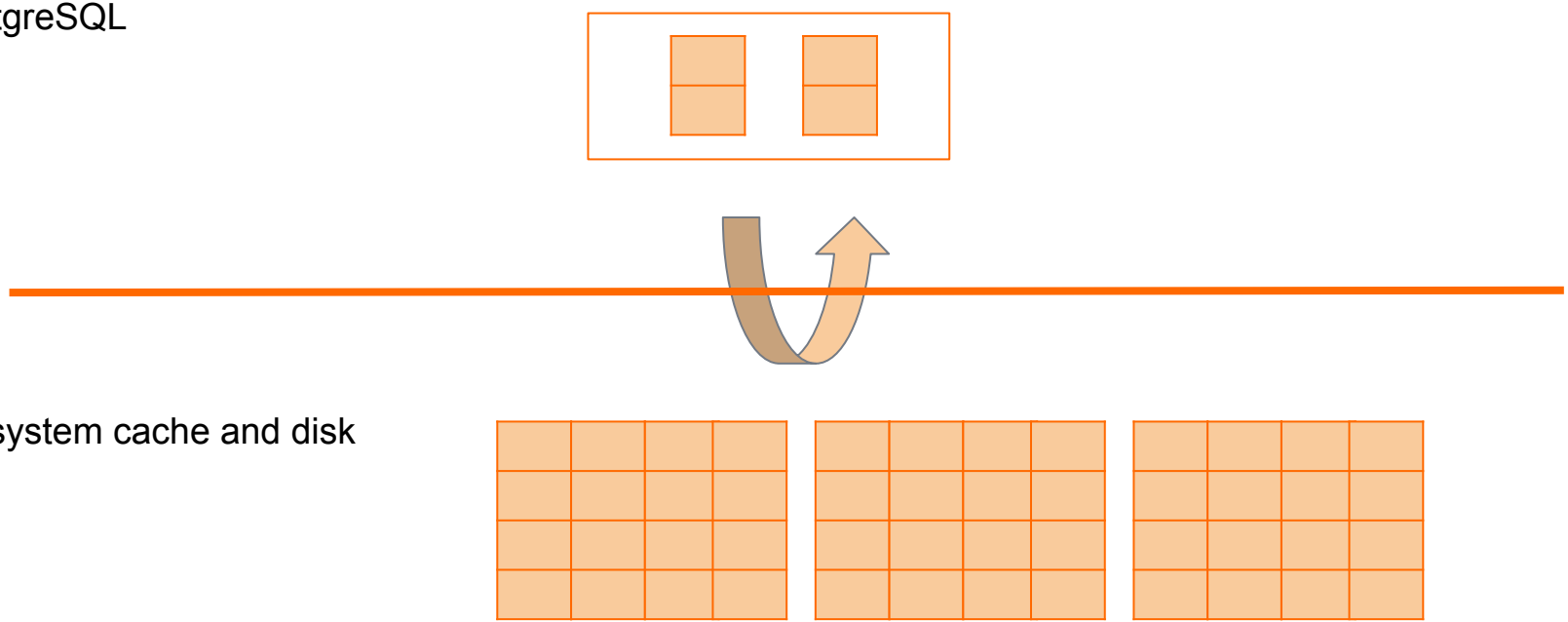
zalando

System’s page



The buffer pool

PostgreSQL



Filesystem cache and disk

zalando

Typical issues

Buffer manager is PostgreSQL.

Filesystem cache is important.

Be careful with load tests.

Typical issues

Use EXPLAIN (ANALYZE, BUFFERS)

Nikolay Samokhvalov, EXPLAIN (ANALYZE) needs BUFFERS
https://tinyurl.com/ywyhpz73

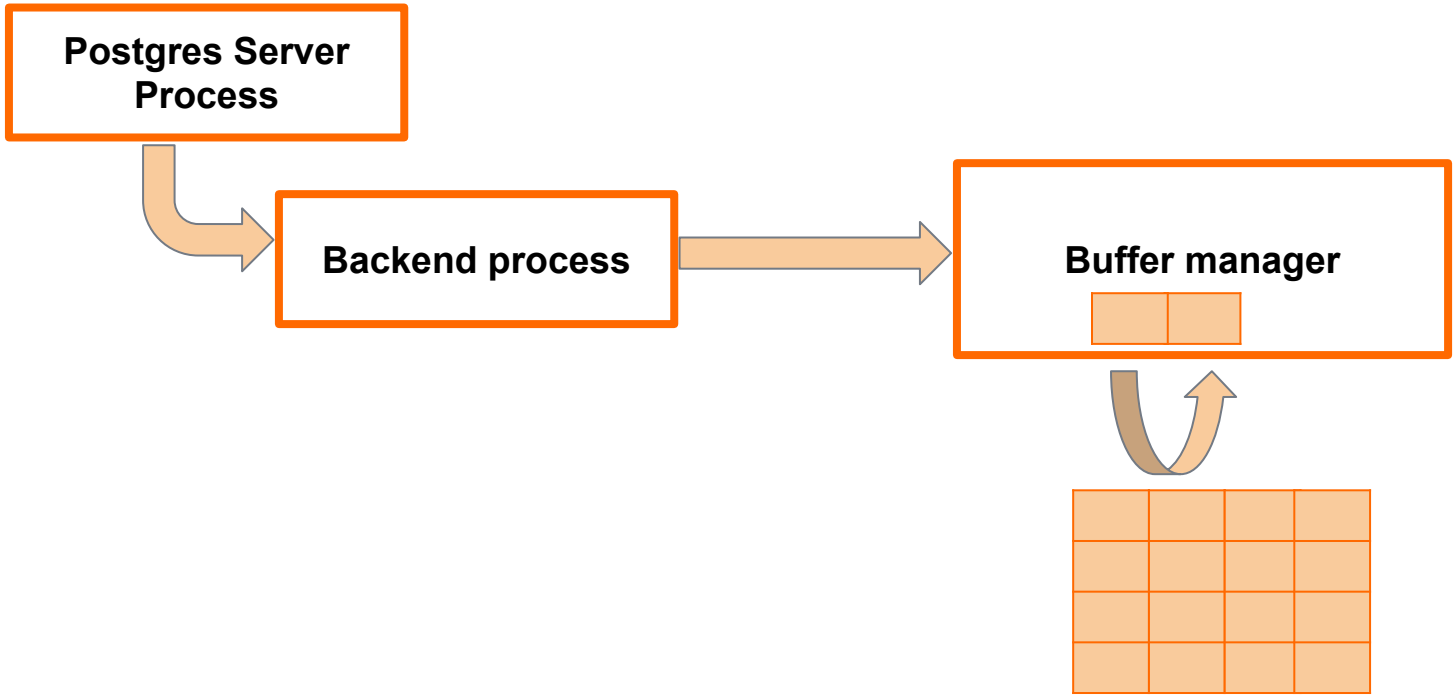
zalando

Typical issues

```
EXPLAIN (ANALYZE, BUFFERS) SELECT * FROM t;
                                   QUERY PLAN
═══════════════════════════════════════════════════════════════════
 Seq Scan on t  (cost=0.00..193460.13 rows=10000113 width=41)
(actual time=0.076..1384.638 rows=10000010.00 loops=1)
 Buffers: shared hit=12659 read=80800
 Planning Time: 3.210 ms
 Execution Time: 1711.777 ms
(4 rows)
```

zalando

Mental model v1.5



zalando

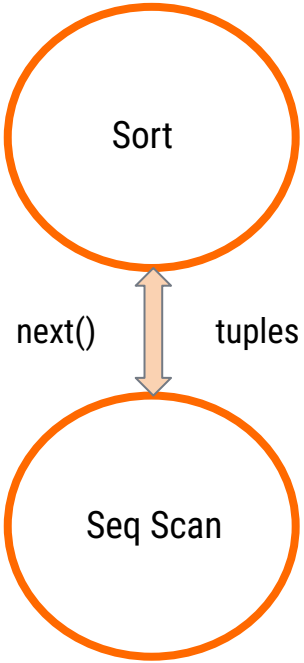
```
SELECT * FROM t ORDER BY payload;
```

Sort and Seq Scan ?

```
EXPLAIN  SELECT * FROM t ORDER BY payload;
                        QUERY PLAN
=================================================================
=================================================
 Sort  (cost=1971392.92..1996393.20 rows=10000113
width=41)
   Sort Key: payload
   ->  Seq Scan on t  (cost=0.00..193460.13 rows=10000113
width=41)
```

zalando

Iterator model

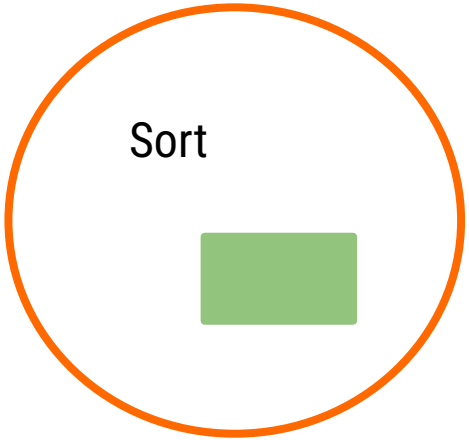


As in Java Iterator<E>:

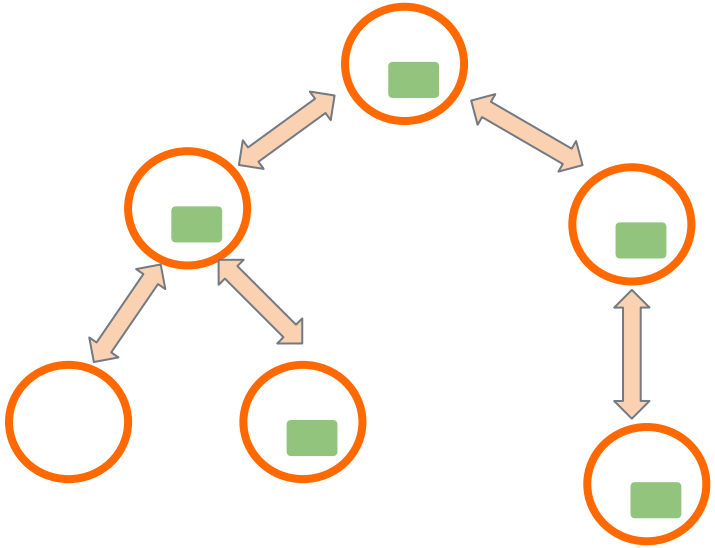
1. next()
2. hasNext()

Goetz Graefe,
Volcano - an Extensible and Parallel Query Evaluation System
https://tinyurl.com/4zaxyjzy

work_mem



Mind the memory !



Reads generate writes

```
EXPLAIN (ANALYZE, BUFFERS) SELECT * FROM t ORDER BY payload;
QUERY PLAN
═══════════════════════════════════════════════════════════
 Sort  (cost=1971392.92..1996393.20 rows=10000113 width=41) (actual
time=34302.764..44241.243 rows=10000010.00 loops=1)
   Sort Key: payload
   Sort Method: external merge  Disk: 499152kB
   Buffers: shared hit=12691 read=80768, temp read=124778 written=124996
   ->  Seq Scan on t  (cost=0.00..193460.13 rows=10000113 width=41) (actual
time=0.145..3214.634 rows=10000010.00 loop>
         Buffers: shared hit=12691 read=80768
 Planning Time: 2.388 ms
 Execution Time: 44699.701 ms
```

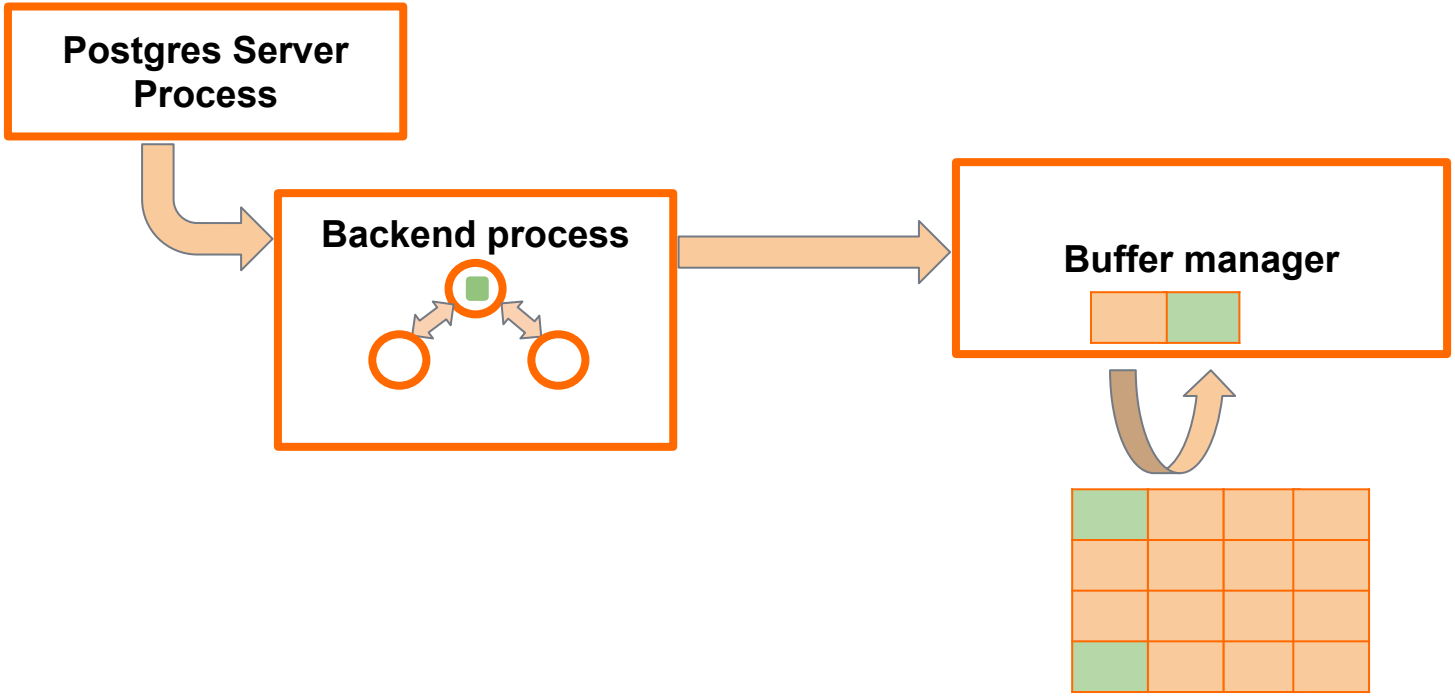
zalando

Reads generate writes

```
SET work_mem = '1000MB';
EXPLAIN (ANALYZE, BUFFERS) SELECT * FROM t ORDER BY payload;
QUERY PLAN
═══════════════════════════════════════════════════════════
 Sort  (cost=1356148.92..1381149.20 rows=10000113 width=41) (actual
time=46419.621..49033.799 rows=10000010.00 loops=1)
   Sort Key: payload
   Sort Method: quicksort  Memory: 904274kB
   Buffers: shared hit=12755 read=80704
   ->  Seq Scan on t  (cost=0.00..193460.13 rows=10000113 width=41) (actual
time=0.033..1302.849 rows=10000010.00 loop>
         Buffers: shared hit=12755 read=80704
 Planning Time: 4.058 ms
 Execution Time: 49732.897 ms
```

zalando

Mental model v2.0



zalando

The plan

How PostgreSQL runs

How PostgreSQL runs SELECTs

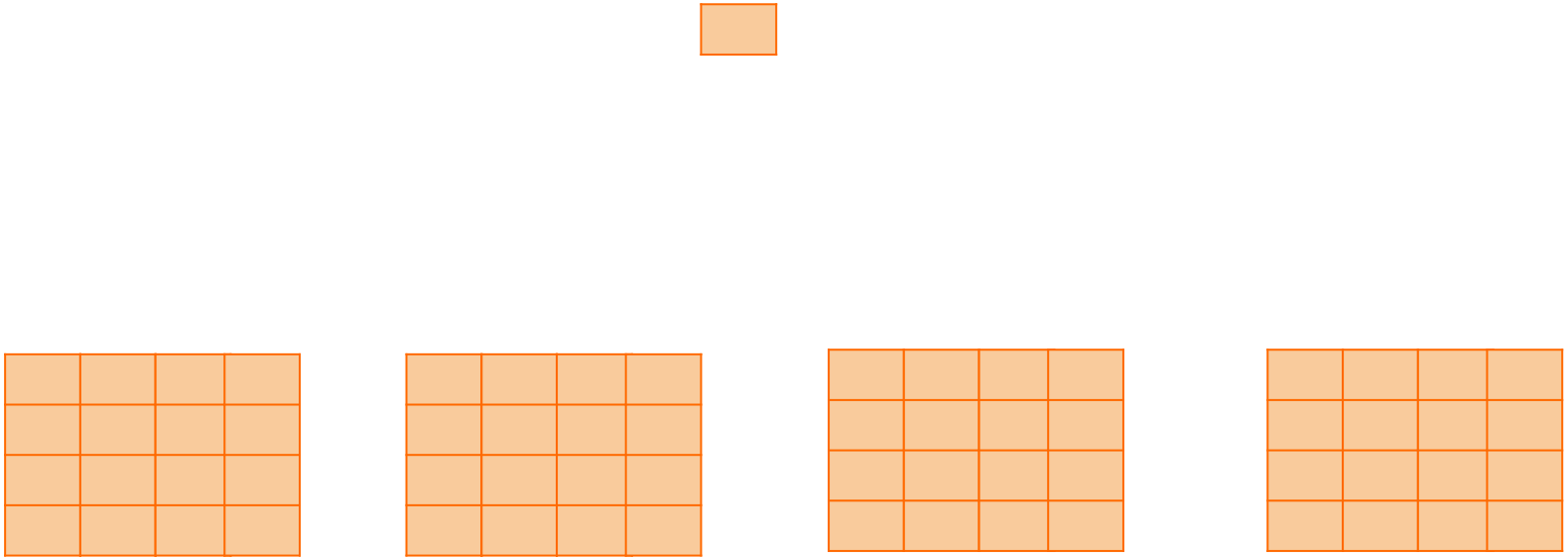
How PostgreSQL runs SELECTs fast

```
SELECT * FROM t
WHERE payload = '...' ;
```

```
EXPLAIN (ANALYZE, BUFFERS) SELECT * FROM t WHERE payload =
'db5c138e-5f56-402b-88d3-fae4d42a7c43';
QUERY PLAN
═══════════════════════════════════════════════════════════════════════
 Gather  (cost=1000.00..146543.02 rows=1 width=41) (actual time=0.496..350.039
rows=1.00 loops=1)
   Workers Planned: 2
   Workers Launched: 2
   Buffers: shared hit=13011 read=80448
   ->  Parallel Seq Scan on t  (cost=0.00..145542.92 rows=1 width=41) (actual
time=226.272..341.485 rows=0.33 loops=3)
         Filter: (payload = 'db5c138e-5f56-402b-88d3-fae4d42a7c43'::text)
         Rows Removed by Filter: 3333336
         Buffers: shared hit=13011 read=80448
 Planning Time: 0.089 ms
 Execution Time: 350.074 ms
```

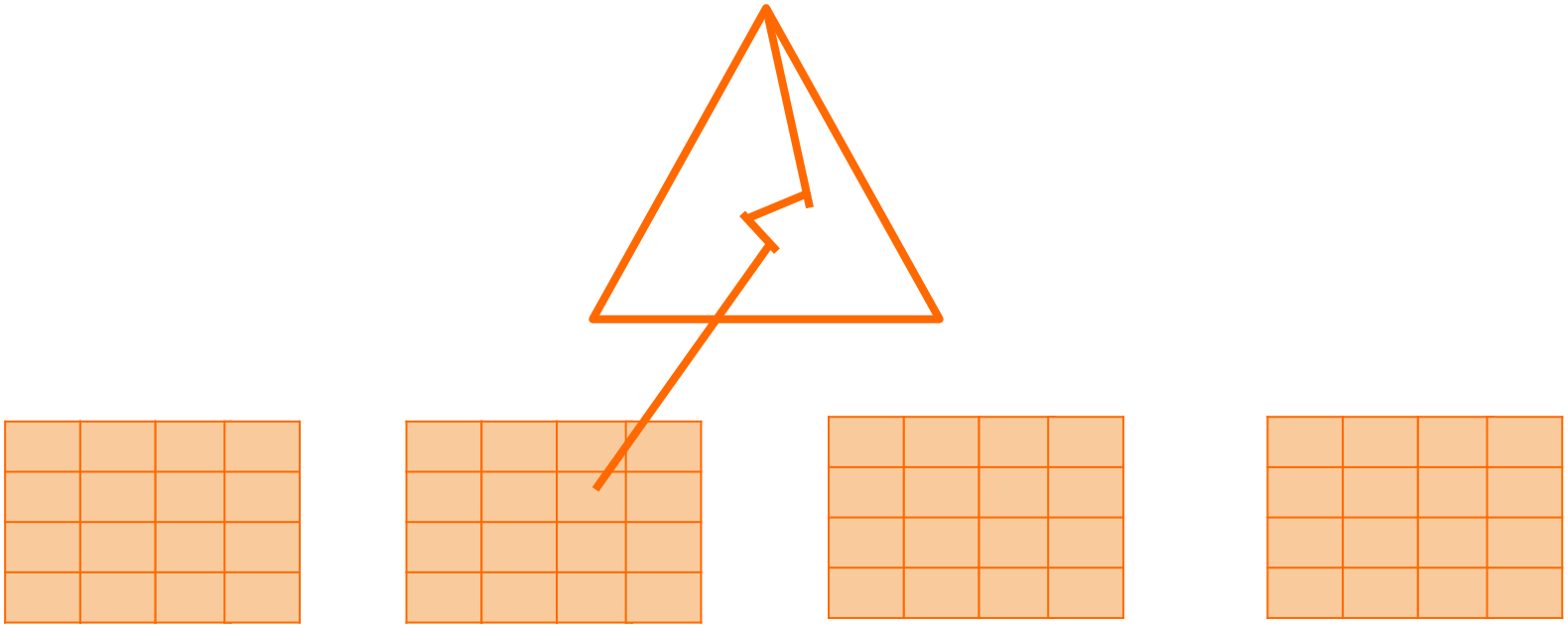
zalando

What a waste …



zalando

How to locate



Aside: B-trees

Fundamental: 99% of all indexes

PostgresPro, Indexes in PostgreSQL, part 4 (B-Tree)
https://tinyurl.com/y9bbjwxm

```
CREATE INDEX CONCURRENTLY ON t(payload);
CREATE INDEX
Time: 66830.423 ms (01:06.830)
```

zalando

How to locate

```
EXPLAIN (ANALYZE, BUFFERS) SELECT * FROM t WHERE payload =
'db5c138e-5f56-402b-88d3-fae4d42a7c43';
QUERY PLAN
═══════════════════════════════════════════════════════════
 Index Scan using t_payload_idx on t  (cost=0.56..8.58 rows=1 width=41)
(actual time=0.617..0.619 rows=1.00 loops=1)
   Index Cond: (payload = 'db5c138e-5f56-402b-88d3-fae4d42a7c43'::text)
   Buffers: shared hit=3 read=2
 Planning:
   Buffers: shared hit=10 read=3 dirtied=1
 Planning Time: 4.571 ms
 Execution Time: 1.386 ms
```

zalando

How to choose

One access method costs more than the other

Index Scan: 5 ms, Seq Scan: 350 ms

Observation: it’s relative

The cost

Cost = IO cost + CPU cost

IO = f(number of pages)

CPU = f(number of tuples)

The statistics

of pages, # of tuples, most common values etc.

Normally collected automatically

Typical issues

Outdated, absent or otherwise wrong statistics

Query optimization on a different dataset

Query plan flips (tip: learn auto_explain)

Almighty ANALYZE

The very first step in query optimization

Before v18: the very first step after major version upgrade

```
ANALYZE t;
ANALYZE
Time: 699.738 ms
```

zalando

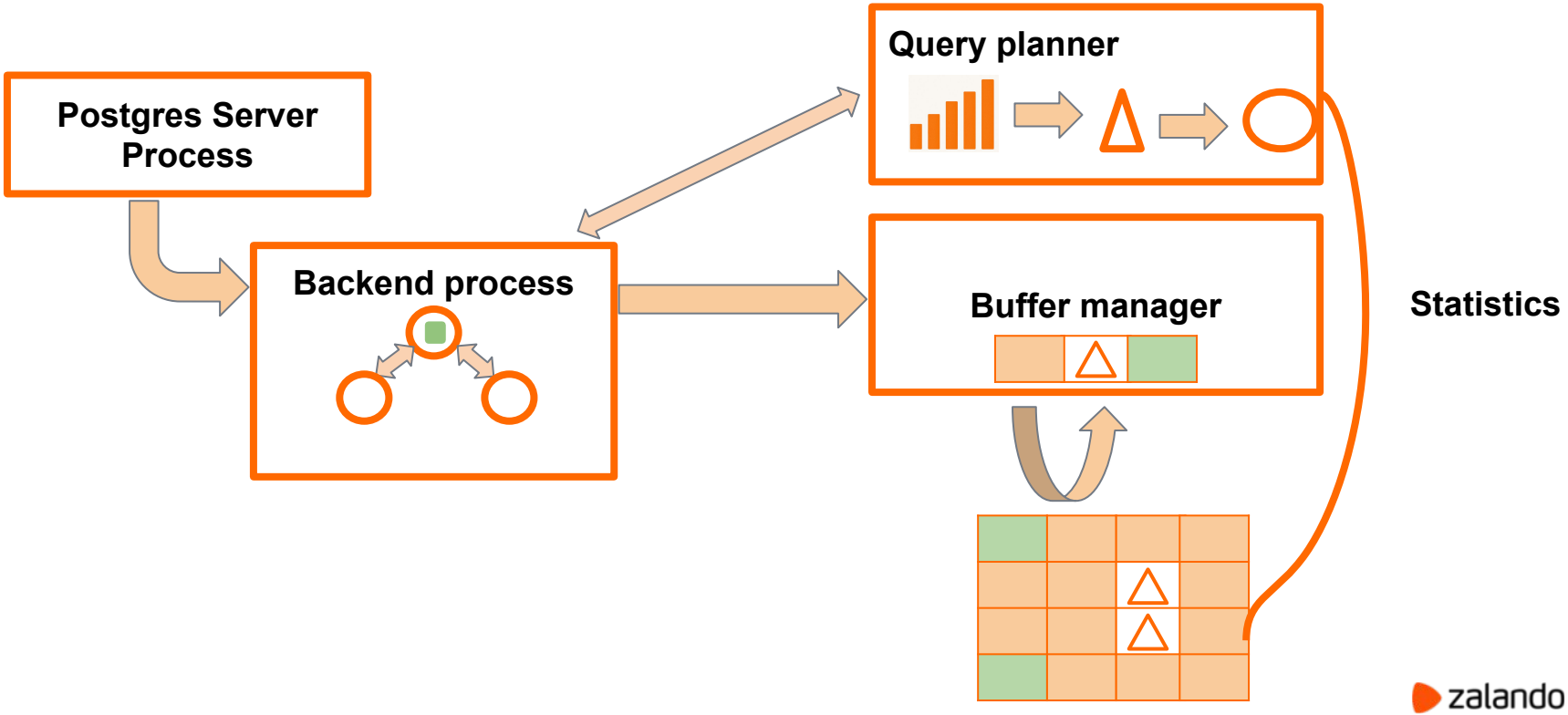
The planner

Statistics

Access methods / joins

Plan / iterator tree

Mental model v3.0



zalando

What’s next ?

Carnegie Mellon Database Group

PEOPLE PROJECTS PUBLICATIONS COURSES NEWS

15-445/645 — Intro to Database Systems

This project-oriented course provides an introduction to the internal architecture of database systems.

6.5830/6.5831: Database Systems

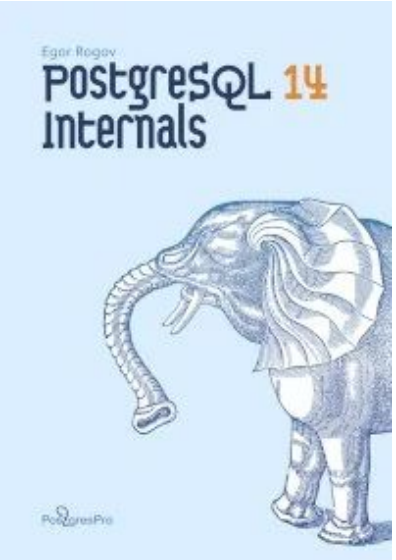
Fall 2024

Home | Schedule | Notes & Assignments | Syllabus | FAQ

+ Implement == understand
- Minimum 150 hours

zalando

But I want my elephants …

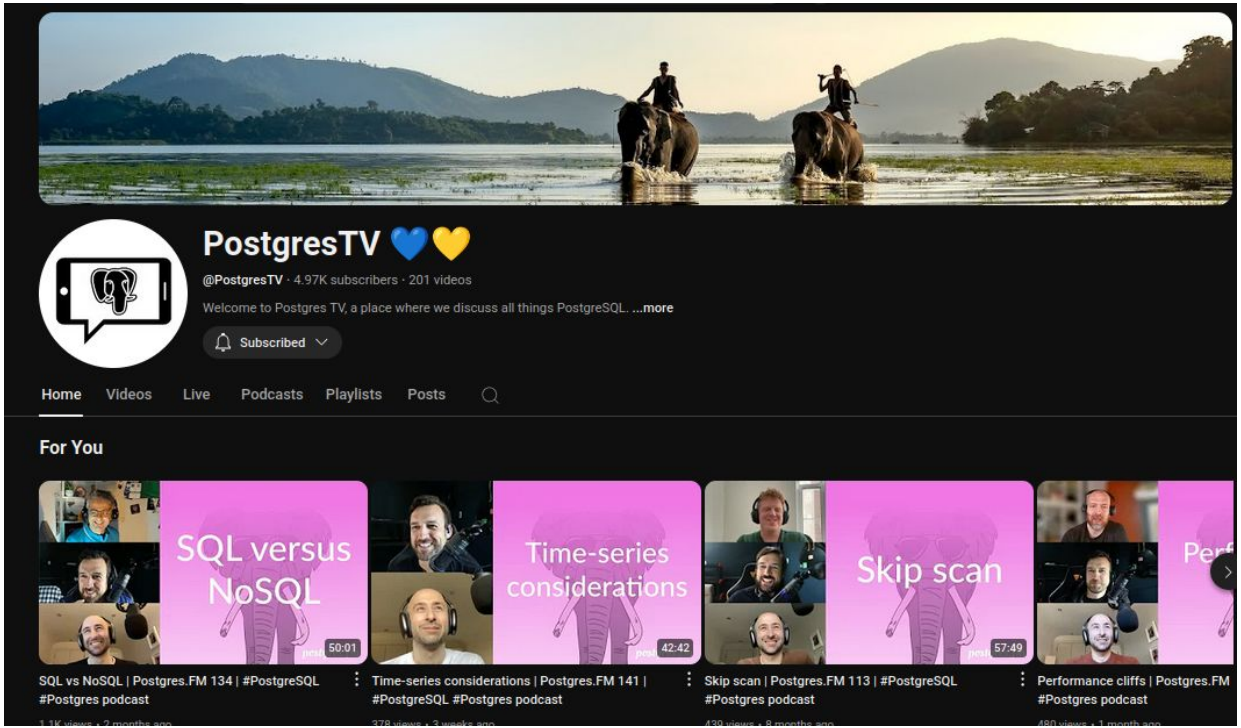


https://www.interdb.jp/pg/index.html

https://postgrespro.com/community/books/internals

zalando

… and fun



zalando

Next talk

The almighty B-trees

Bonus

```
SELECT payload FROM t WHERE id = 1;
              payload
-----------------------------------------
 133b4937-9211-4add-9839-749955xfa2da
(1 row)
```

zalando

Bonus

```
SELECT payload::uuid FROM t WHERE id = 1;

ERROR:  22P02: invalid input syntax for type uuid:
"133b4937-9211-4add-9839-749955xfa2da"
LOCATION:  string_to_uuid, uuid.c:170
Time: 0.559 ms
```

zalando

Bonus

```
CREATE TABLE t (
        id SERIAL PRIMARY KEY,
        payload TEXT
);

INSERT INTO t (payload)
SELECT gen_random_uuid()::TEXT
FROM generate_series(1, 10000000);
```

Not everything is TEXT

UTFT

Use
The
F*bulous
Typesystem

zalando

Thank You!

zalando