PostgreSQL Conference
Germany 2026

Jonas Gassenmeyer

Ein Jahr PostgreSQL statt Oracle - Das Leben danach

- jonas.gassenmeyer@deutschebahn.com
- Ein Herz für relationale Datenbanken
- Nervös, weil ich ein bisschen aus dem “Konferenz-Tritt” bin

Die nächsten ~~45~~ 60 Minuten

Übertrage ~~Wissen~~ **einfach** von Oracle nach PostgreSQL

Was soll schon schief gehen?

Dann lasst uns loslegen…

```
create table demo (
 descr varchar2(10),
 col varchar2(10)
);
```

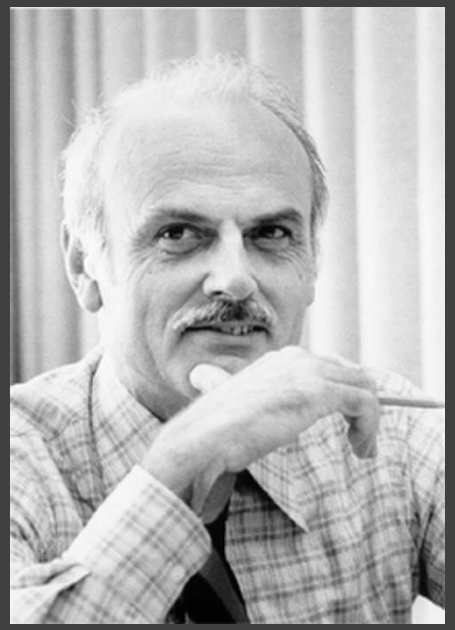

```
SQL Error [42704]: ERROR:
type "varchar2" does not
exist
```

```
create table demo (
 descr text,
 col text
);
```

```
select *
  from all_tables
 where table_name = 'demo'
;
```

```
select *
  from all_tables
 where table_name = 'demo'
;
```

```
select *
  from pg_class
 where relname = 'demo'
;
```



```
insert into demo (descr, col) values ('1 insert', '');
insert into demo (descr, col) values ('2 insert', null);
```

```
insert into demo (descr, col) values ('1 insert', '');
insert into demo (descr, col) values ('2 insert', null);
```

```
select *
  from demo
 where col is null;
```

```
insert into demo (descr, col) values ('1 insert', '');
insert into demo (descr, col) values ('2 insert', null);
```

```
select *
  from demo
 where col is null;

|descr      | col
+-----------+------------
|2 insert   | (null)
```

```
... WHERE coalesce(col, '') = ''
```

https://aws.amazon.com/blogs/database/handle-empty-strings-when-migrating-from-oracle-to-postgresql/

```
... WHERE coalesce(col, '') = ''
```

https://aws.amazon.com/blogs/database/handle-empty-strings-when-migrating-from-oracle-to-postgresql/

```
... WHERE coalesce(col, '') = ''
```

13 Word pages (3300 words)

https://aws.amazon.com/blogs/database/handle-empty-strings-when-migrating-from-oracle-to-postgresql/

```
select 'test'||null from dual;
```


```
SQL Error [42P01]: ERROR:
relation "dual" does not
exist
```

```
select 'test'||null ;
```

```
select 'test'||null ;

|?colum
+------
|(null)
```

```
select '' = '' tst;

|tst
+------------
|true
```

Boolean

```
alter table demo add primary key (descr);


insert into demo values ('key1', 'i repeat');
insert into demo values ('key1', 'i repeat');
```

```
SQL Error [1] [23000]: ORA-00001: unique
constraint (SYSTEM.SYS_C0012482) violated
```

```
SQL Error [23505]: ERROR: duplicate key value violates
unique constraint "demo_pkey"

Detail: Key (descr)=(key1) already exists.
```

```
SQL Error [1] [23000]: ORA-00001: unique
constraint (SYSTEM.SYS_C0012482) violated
```

DB Systel GmbH

MERGE

https://loke.dev/blog/postgres-merge-concurrency-locking-traps

coding without RTFM

https://twitter.com/comrad_pogaca/status/1483463570109403140

```
insert into demo values ('key2', 'i repeat')
on conflict (descr)
do nothing;


insert into demo values ('key2', 'i repeat')
on conflict (descr) --pk or uk
do update set col = 'i''ve been updated';
```

```
select *
  from demo
 where descr = 'key2';


|descr|col
+-----+------------------
|key2 |i've been updated
```

Erstmal neue Testdaten

Website, Aufruf, Dauer

	url	created_at	request_duration
1	3_wbesite_url	2022-08-19 00:00:00	00:00:00.053419
2	2_wbesite_url	2022-08-18 00:00:00	00:00:16.175117
3	5_wbesite_url	2022-08-17 00:00:00	00:00:04.737701
4	10_wbesite_url	2022-08-16 00:00:00	00:00:37.549743
5	8_wbesite_url	2022-08-15 00:00:00	00:00:13.432602
6	1_wbesite_url	2022-08-14 00:00:00	00:00:18.229272
7	4_wbesite_url	2022-08-13 00:00:00	00:00:10.05484
8	6_wbesite_url	2022-08-12 00:00:00	00:00:49.272442
9	3_wbesite_url	2022-08-11 00:00:00	00:00:25.686225
10	6_wbesite_url	2022-08-10 00:00:00	00:00:20.192684
11	3_wbesite_url	2022-08-09 00:00:00	00:00:01.389349
12	9_wbesite_url	2022-08-08 00:00:00	00:00:32.105082
13	8_wbesite_url	2022-08-07 00:00:00	00:00:47.395415
14	5_wbesite_url	2022-08-06 00:00:00	00:00:53.359661
15	8_wbesite_url	2022-08-05 00:00:00	00:00:46.10167
16	3_wbesite_url	2022-08-04 00:00:00	00:00:27.322253
17	6_wbesite_url	2022-08-03 00:00:00	00:00:41.657576
18	6_wbesite_url	2022-08-02 00:00:00	00:00:27.917884
19	6_wbesite_url	2022-08-01 00:00:00	00:00:03.175308

```
select level
from dual
connect by level <= 10;
```


```
SQL Error [42601]: ERROR: syntax error at or near "by"
```

```
select generate_series(1,10) ;
```

Aber das ist ja nicht alles...

```
create table logs as

with t as (
select generate_series('2022-07-01 00:00:00'::timestamp
      , '2022-08-19 01:00:00'::timestamp, '1 day') ts


)
, rn as(
select row_number() over(order by random()) r
     , t.ts
 from t
)
, u as (
select generate_series(1, 10) || '_wbesite_url' url


)

select u.url, rn.ts created_at
     , random() * interval '1 minute' request_duration
from rn
join u
on mod(r,10) +1 =
replace(substring(u.url from 1 for 2), '_', '')::bigint
;
```


```
create table logs as (
select * from (
with t as (
select to_date('2022-07-01 00:00:00'
           , 'YYYY-MM-DD HH24:MI:SS') + level ts
  from dual
 connect by level <= 50
)
, rn as(
select row_number() over(order by dbms_random.value) r
     , t.ts
 from t
)
, u as (
select level || '_wbesite_url' url
  from dual
 connect by level <= 50
)

select u.url, rn.ts created_at
     , dbms_random.value * interval '1' minute request_duration
from rn
join u
on mod(r,10) +1 =
replace(substr(u.url, 1, 2), '_'))
)
;
```

```
create table logs as

with t as (
select generate_series('2022-07-01 00:00:00'::timestamp
      , '2022-08-19 01:00:00'::timestamp, '1 day') ts


)
, rn as(
select row_number() over(order by random()) r
    , t.ts
 from t
)
, u as (
select generate_series(1, 10) || '_wbesite_url' url


)

select u.url, rn.ts created_at
    , random() * interval '1 minute' request_duration
from rn
join u
on mod(r,10) +1 =
replace(substring(u.url from 1 for 2), '_', '')::bigint
;
```

```
create table logs as (
select * from (
with t as (
select to_date('2022-07-01 00:00:00'
           , 'YYYY-MM-DD HH24:MI:SS') + level ts
  from dual
 connect by level <= 50
)
, rn as(
select row_number() over(order by dbms_random.value) r
    , t.ts
 from t
)
, u as (
select level || '_wbesite_url' url
  from dual
 connect by level <= 50
)

select u.url, rn.ts created_at
    , dbms_random.value * interval '1' minute request_duration
from rn
join u
on mod(r,10) +1 =
replace(substr(u.url, 1, 2), '_'))
)
;
```

```
create table logs as

with t as (
select generate_series('2022-07-01 00:00:00'::timestamp
      , '2022-08-19 01:00:00'::timestamp, '1 day') ts


)
, rn as(
select row_number() over(order by random()) r
     , t.ts
 from t
)
, u as (
select generate_series(1, 10) || '_wbesite_url' url


)

select u.url, rn.ts created_at
     , random() * interval '1 minute' request_duration
from rn
join u
on mod(r,10) +1 =
replace(substring(u.url from 1 for 2), '_', '')::bigint
;
```

```
create table logs as (
select * from (
with t as (
select to_date('2022-07-01 00:00:00'
           , 'YYYY-MM-DD HH24:MI:SS') + level ts
  from dual
 connect by level <= 50
)
, rn as(
select row_number() over(order by dbms_random.value) r
     , t.ts
 from t
)
, u as (
select level || '_wbesite_url' url
  from dual
 connect by level <= 50
)

select u.url, rn.ts created_at
     , dbms_random.value * interval '1' minute request_duration
from rn
join u
on mod(r,10) +1 =
replace(substr(u.url, 1, 2), '_'))
)
;
```

```
create table logs as

with t as (
select generate_series('2022-07-01 00:00:00'::timestamp
      , '2022-08-19 01:00:00'::timestamp, '1 day') ts



)
, rn as(
select row_number() over(order by random()) r
     , t.ts
 from t
)
, u as (
select generate_series(1, 10) || '_wbesite_url' url



)

select u.url, rn.ts created_at
     , random() * interval '1 minute' request_duration
from rn
join u
on mod(r,10) +1 =
replace(substring(u.url from 1 for 2), '_', '')::bigint
;
```

```
create table logs as (
select * from (
with t as (
select to_date('2022-07-01 00:00:00'
           , 'YYYY-MM-DD HH24:MI:SS') + level ts
  from dual
 connect by level <= 50
)
, rn as(
select row_number() over(order by dbms_random.value) r
     , t.ts
 from t
)
, u as (
select level || '_wbesite_url' url
  from dual
 connect by level <= 50
)

select u.url, rn.ts created_at
     , dbms_random.value * interval '1' minute request_duration
from rn
join u
on mod(r,10) +1 =
replace(substr(u.url, 1, 2), '_'))
)
;
```

```
create table logs as

with t as (
select generate_series('2022-07-01 00:00:00'::timestamp
      , '2022-08-19 01:00:00'::timestamp, '1 day') ts



)
, rn as(
select row_number() over(order by random()) r
     , t.ts
 from t
)
, u as (
select generate_series(1, 10) || '_wbesite_url' url



)

select u.url, rn.ts created_at
     , random() * interval '1 minute' request_duration
from rn
join u
on mod(r,10) +1 =
replace(substring(u.url from 1 for 2), '_', '')::bigint
;
```

```
create table logs as (
select * from (
with t as (
select to_date('2022-07-01 00:00:00'
           , 'YYYY-MM-DD HH24:MI:SS') + level ts
  from dual
 connect by level <= 50
)
, rn as(
select row_number() over(order by dbms_random.value) r
     , t.ts
 from t
)
, u as (
select level || '_wbesite_url' url
  from dual
 connect by level <= 50
)

select u.url, rn.ts created_at
     , dbms_random.value * interval '1' minute request_duration
from rn
join u
on mod(r,10) +1 =
replace(substr(u.url, 1, 2), '_'))
)
;
```

Aggregieren...z.B. Quotienten bilden...

```
select (...) / count(distinct url) over()
  from logs
;
```


```
SQL Error [0A000]: ERROR: DISTINCT is not
implemented for window functions
```

Weil wir gerade von DISTINCT sprechen...

Für jede Website nur den letzten Logeintrag und dessen request_duration?

	url	created_at	request_duration
1	3_wbesite_url	2022-08-19 00:00:00	00:00:00.053419
2	2_wbesite_url	2022-08-18 00:00:00	00:00:16.175117
3	5_wbesite_url	2022-08-17 00:00:00	00:00:04.737701
4	10_wbesite_url	2022-08-16 00:00:00	00:00:3[illegible]549743
5	8_wbesite_url	2022-08-15 00:00:00	00:00:13.432602
6	1_wbesite_url	2022-08-14 00:00:00	00:00:18.229272
7	4_wbesite_url	2022-08-13 00:00:00	00:00:10.05484
8	6_wbesite_url	2022-08-12 00:00:00	00:00:49.272442
9	3_wbesite_url	2022-08-11 00:00:00	00:00:25.686225
10	6_wbesite_url	2022-08-10 00:00:00	00:00:20.192684
11	3_wbesite_url	2022-08-09 00:00:00	00:00:01.389349
12	9_wbesite_url	2022-08-08 00:00:00	00:00:32.105082
13	8_wbesite_url	2022-08-07 00:00:00	00:00:47.395415
14	5_wbesite_url	2022-08-06 00:00:00	00:00:53.359661
15	8_wbesite_url	2022-08-05 00:00:00	00:00:46.10167
16	3_wbesite_url	2022-08-04 00:00:00	00:00:27.322253
17	6_wbesite_url	2022-08-03 00:00:00	00:00:41.657576
18	6_wbesite_url	2022-08-02 00:00:00	00:00:27.917884
19	6_wbesite_url	2022-08-01 00:00:00	00:00:03.175308

```
select url
     , max(request_duration) keep(dense_rank last order by created_at)
     , max(created_at)
  from logs
 group by url
;
```

DB Systel GmbH

```
select distinct on (url) url, request_duration
  from logs
 order by url, created_at desc
;
```

https://www.geekytidbits.com/postgres-distinct-on/

```
select distinct on (url) url, request_duration
  from logs
 order by url, created_at desc
;
```

https://www.geekytidbits.com/postgres-distinct-on/

Arithmetik

```
select 5/2;
```

```
select 5/2;

|?column?
+--------
|        2
```

...Kein Formatierungsproblem...WYSIWYG

numeric_type **/** ***numeric_type*** → ***numeric_type***

Division (for integral types, division truncates the result towards zero)

```
5.0 / 2 → 2.5000000000000000
5 / 2 → 2
(-5) / 2 → -2
```

https://www.postgresql.org/docs/14/functions-math.html

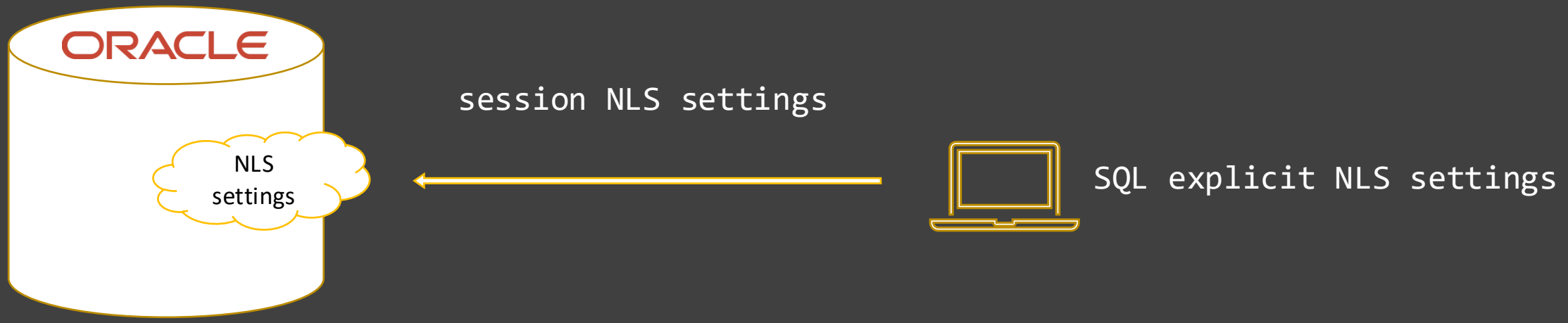
```
select 5.0/2;

select 5::float/2

select pg_typeof(5.0/2)
```


WHAT TIME IS IT

9 vs 6 fractional seconds


ORACLE
NLS
settings
session NLS settings
SQL explicit NLS settings


IANA
UTC!
set timezone = 'Europe/Berlin';
select now() at time zone 'UTC'

TIMESTAMPTZ

*“no such thing as a "server timezone" (...)The server does have a "log timezone" setting, but that's for log messages(..)The string you send to the server is assumed to be in the timezone defined by the client-controllable timezone setting, and it is **converted to UTC for storage**. (Of course, if the string you send contains an explicit zone name or UTC offset, we believe that instead; but in any case the bits on disk represent a time in UTC.)”*

https://www.postgresql.org/message-id/CADH0_2XcdBxDJMqa5q59gnPzpZO_f4E6A034R1%2B-G21-ByB2_w%40mail.gmail.com

https://www.cybertec-postgresql.com/en/time-zone-management-in-postgresql/

tsrange

multiranges (v14)

temporal constraints (v18)

https://www.cybertec-postgresql.com/en/multiranges-in-postgresql-14/

https://hashrocket.com/blog/posts/postgresql-18-temporal-constraints

Genug Syntax...

Lasst uns unter die Haube schauen

user != schema

Concurrent sessions: MVCC

MVCC

- Oracle avoids moving rows at all price
- PostgreSQL is doing something like a Copy-On-Write

https://franckpachot.medium.com/mvcc-in-oracle-vs-postgres-and-a-little-no-bloat-beauty-e3a04d6a8bd2

Bloat


WHAT ARE YOU DOING?
CLEAN UP IMMEDIATELY.
memegenerator.net

That’s the vacuum process

That’s the vacuum process


```
vacuum threshold =
    vacuum base threshold +
        (vacuum scale factor
            *  number of tuples)
```

when 20% of the rows plus 50 rows are inserted, updated, or deleted

```
postgreSQL>cat postgresql.conf | grep vacuum
#autovacuum_work_mem = -1                # min 1MB, or -1 to use maintenance_work_mem
#vacuum_cost_delay = 0                   # 0-100 milliseconds (0 disables)
#vacuum_cost_page_hit = 1                # 0-10000 credits
#vacuum_cost_page_miss = 2               # 0-10000 credits
#vacuum_cost_page_dirty = 20             # 0-10000 credits
#vacuum_cost_limit = 200                 # 1-10000 credits
#vacuum_defer_cleanup_age = 0    # number of xacts by which cleanup is delayed
#log_autovacuum_min_duration = -1        # log autovacuum activity;
#autovacuum = on                         # Enable autovacuum subprocess?  'on'
#autovacuum_max_workers = 3              # max number of autovacuum subprocesses
#autovacuum_naptime = 1min               # time between autovacuum runs
#autovacuum_vacuum_threshold = 50        # min number of row updates before
                                         # vacuum
#autovacuum_vacuum_insert_threshold = 1000      # min number of row inserts
                                         # before vacuum; -1 disables insert
                                         # vacuums
#autovacuum_analyze_threshold = 50       # min number of row updates before
#autovacuum_vacuum_scale_factor = 0.2    # fraction of table size before vacuum
#autovacuum_vacuum_insert_scale_factor = 0.2    # fraction of inserts over table
                                         # size before insert vacuum
#autovacuum_analyze_scale_factor = 0.1   # fraction of table size before analyze
#autovacuum_freeze_max_age = 200000000   # maximum XID age before forced vacuum
#autovacuum_multixact_freeze_max_age = 400000000        # maximum multixact age
                                         # before forced vacuum
#autovacuum_vacuum_cost_delay = 2ms      # default vacuum cost delay for
                                         # autovacuum, in milliseconds;
                                         # -1 means use vacuum_cost_delay
#autovacuum_vacuum_cost_limit = -1       # default vacuum cost limit for
                                         # autovacuum, -1 means use
                                         # vacuum_cost_limit
#vacuum_freeze_table_age = 150000000
#vacuum_freeze_min_age = 50000000
#vacuum_failsafe_age = 1600000000
#vacuum_multixact_freeze_table_age = 150000000
#vacuum_multixact_freeze_min_age = 5000000
#vacuum_multixact_failsafe_age = 1600000000
```

10-things-i-hate-about-postgresql

https://www.cybertec-postgresql.com/en/transaction-id-wraparound-a-walk-on-the-wild-side/

Weil wir gerade über Transaktionen sprechen...

- Tools im Default: auto commit
- DDL ist ebenfalls Transaktions-sicher (rollback)
- `PRAGMA AUTONOMOUS_TRANSACTION` existiert nicht (PL/SQL)

Performance

```
explain plan set statement_id = 'ex_plan2' for
select (...)


select plan_table_output
  from table(dbms_xplan.display(null, 'ex_plan2' ,'typical'));
```

```
explain plan set statement_id = 'ex_plan2' for
select (...)


select plan_table_output
  from table(dbms_xplan.display(null, 'ex_plan2' ,'typical'));
```

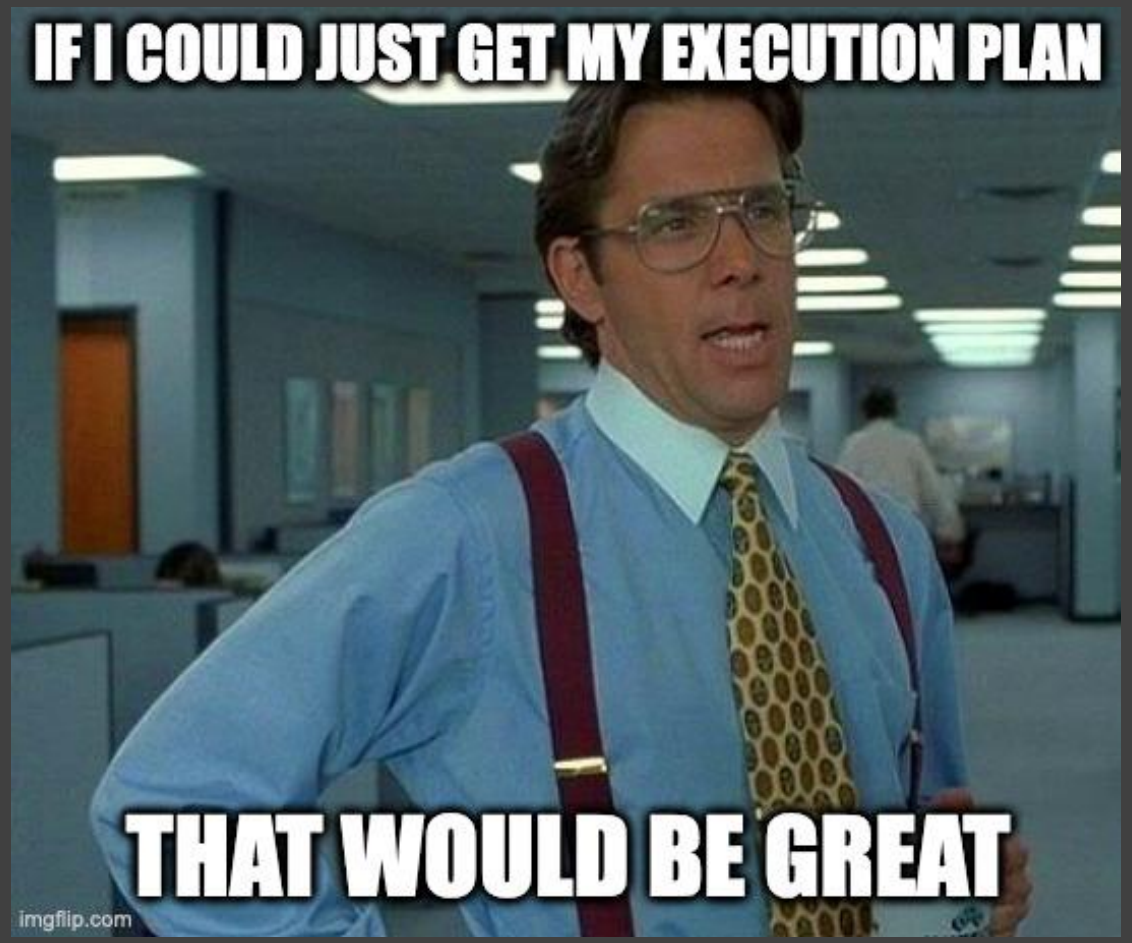
```
'TYPICAL +PEEKED_BINDS'
```

```
explain plan set statement_id = 'ex_plan2' for
select (...)


select plan_table_output
  from table(dbms_xplan.display(null, 'ex_plan2' ,'typical'));
```

```
'TYPICAL +PEEKED_BINDS'

'BASIC'
```


IF I COULD JUST GET MY EXECUTION PLAN
THAT WOULD BE GREAT
imgflip.com

```
explain (analyze, buffers)
select * from demo;
```

https://www.cybertec-postgresql.com/en/how-to-interpret-postgresql-explain-analyze-output

Query wird auch wirklich ausgeführt

```
explain (analyze, buffers)
select * from demo;
```

I/O + Buffers, Default in v18

```
explain (analyze, buffers)
select * from demo;
```

```
Gather  (cost=1209.94..958085.43 rows=4642 width=219) (actual time=1109.578..58769.059 rows=1590 loops=1)
  Workers Planned: 2
  Workers Launched: 2
  Buffers: shared hit=15078 read=642417
  ->  Hash Left Join  (cost=209.94..956621.23 rows=1934 width=219) (actual time=1650.670..58733.230 rows=530 loops=3)
        Hash Cond: ((r.vehicle_id)::text = (v.id)::text)
        Buffers: shared hit=15078 read=642417
        ->  Nested Loop Left Join  (cost=1.39..956339.89 rows=1934 width=394) (actual time=1644.500..58722.864 rows=530 loops=3)
              Buffers: shared hit=14573 read=642321
              ->  Nested Loop  (cost=0.83..915084.87 rows=1934 width=370) (actual time=1643.029..58215.242 rows=530 loops=3)
                    Buffers: shared hit=9596 read=640105
                    ->  Nested Loop  (cost=0.27..815869.64 rows=2004 width=362) (actual time=1640.491..57383.195 rows=530 loops=3)
                          Join Filter: (r.created_at >= ((now())::date - ((p.prebooking_max_threshold)::double precision * '00:00:01'::interval)))
                          Rows Removed by Join Filter: 0
                          Buffers: shared hit=4916 read=636702
                          ->  Parallel Seq Scan on rides r  (cost=0.00..765918.23 rows=6011 width=389) (actual time=1639.946..57375.890 rows=530 loops=3)
                                Filter: (((product_id)::text = 'prd_7fff391f-463a-4793-86e9-ea3574d56a6d'::text) AND ((created_at + ((prebooking_duration)::double precision * '00:00:01'::interval)) >
                                Rows Removed by Filter: 2197905
                                Buffers: shared hit=143 read=636700
                          ->  Index Scan using products_pkey on products p  (cost=0.27..8.29 rows=1 width=46) (actual time=0.009..0.009 rows=1 loops=1591)
                                Index Cond: ((id)::text = 'prd_7fff391f-463a-4793-86e9-ea3574d56a6d'::text)
                                Buffers: shared hit=4773 read=2
                    ->  Index Scan using index_requested_points_on_ride_id on requested_points rp  (cost=0.56..49.44 rows=7 width=49) (actual time=1.525..1.567 rows=1 loops=1590)
                          Index Cond: ((ride_id)::text = (r.id)::text)
                          Filter: ((waypoint_type)::text = 'origin'::text)
                          Rows Removed by Filter: 1
                          Buffers: shared hit=4680 read=3403
Planning:
  Buffers: shared hit=1145 read=197
Planning Time: 125.755 ms
JIT:
  Functions: 96
  Options: Inlining true, Optimization true, Expressions true, Deforming true
  Timing: Generation 17.975 ms, Inlining 296.459 ms, Optimization 1292.804 ms, Emission 791.845 ms, Total 2399.083 ms
Execution Time: 58846.312 ms
```

```
Gather  (cost=1209.94..958085.43 rows=4642 width=219) (actual time=1109.578..58769.059 rows=1590 loops=1)
  Workers Planned: 2
  Workers Launched: 2
  Buffers: shared hit=15078 read=642417
  ->  Hash Left Join  (cost=209.94..956621.23 rows=1934 width=219) (actual time=1650.670..58733.230 rows=530 loops=3)
        Hash Cond: ((r.vehicle_id)::text = (v.id)::text)
        Buffers: shared hit=15078 read=642417
        ->  Nested Loop Left Join  (cost=1.39..956339.89 rows=1934 width=394) (actual time=1644.500..58722.864 rows=530 loops=3)
              Buffers: shared hit=14573 read=642321
              ->  Nested Loop  (cost=0.83..915084.87 rows=1934 width=370) (actual time=1643.029..58215.242 rows=530 loops=3)
                    Buffers: shared hit=9596 read=640105
                    ->  Nested Loop  (cost=0.27..815869.64 rows=2004 width=362) (actual time=1640.491..57383.195 rows=530 loops=3)
                          Join Filter: (r.created_at >= ((now())::date - ((p.prebooking_max_threshold)::double precision * '00:00:01':::interval)))
                          Rows Removed by Join Filter: 0
                          Buffers: shared hit=4916 read=636702
                          ->  Parallel Seq Scan on rides r  (cost=0.00..765918.23 rows=6011 width=389) (actual time=1639.946..57375.890 rows=530 loops=3)
                                Filter: (((product_id)::text = 'prd_7fff391f-463a-4793-86e9-ea3574d56a6d'::text) AND ((created_at + ((prebooking_duration)::double precision * '00:00:01':::interval)) >
                                Rows Removed by Filter: 2197905
                                Buffers: shared hit=143 read=636700
                          ->  Index Scan using products_pkey on products p  (cost=0.27..8.29 rows=1 width=46) (actual time=0.009..0.009 rows=1 loops=1591)
                                Index Cond: ((id)::text = 'prd_7fff391f-463a-4793-86e9-ea3574d56a6d'::text)
                                Buffers: shared hit=4773 read=2
                    ->  Index Scan using index_requested_points_on_ride_id on requested_points rp  (cost=0.56..49.44 rows=7 width=49) (actual time=1.525..1.567 rows=1 loops=1590)
                          Index Cond: ((ride_id)::text = (r.id)::text)
                          Filter: ((waypoint_type)::text = 'origin'::text)
                          Rows Removed by Filter: 1
                          Buffers: shared hit=4680 read=3403
Planning:
  Buffers: shared hit=1145 read=197
Planning Time: 125.755 ms
JIT:
  Functions: 96
  Options: Inlining true, Optimization true, Expressions true, Deforming true
  Timing: Generation 17.975 ms, Inlining 296.459 ms, Optimization 1292.804 ms, Emission 791.845 ms, Total 2399.083 ms
Execution Time: 58846.312 ms
```

Tools?

- Instrumentation muss explizit angeschaltet werden
- Kein raw tracing einer Session

Tools?

- (kein) sampling im ASH-Stil (`pg_sentinel`)
- (keine) Hints & baselines (`pg_hint_plan`)
- Extensions statt GRANTs ;-)

Tools?

pg_ash (Feb 2026)

- (kein) sampling im ASH-Stil (`pg_sentinel`)
- (keine) Hints & baselines (`pg_hint_plan`)
- Extensions statt GRANTs ;-)

Tools?

- (kein) sampling im ASH-Stil (`pg_sentinel`)
- (keine) Hints & baselines (`pg_hint_plan`)
- Extensions statt GRANTs ;-)

pg_stat_statements

generelle, nachträgliche Analyse von schlechten SQLs

(keine Literale -> Bind Variablen)

auto_explain

detaillierte nachträgliche Analyse **eines spezielles SQLs**

(Zugriff auf Server logs)

pg_show_plans

“hands on” **während** langsame Query ausgeführt wird (zweite Session)

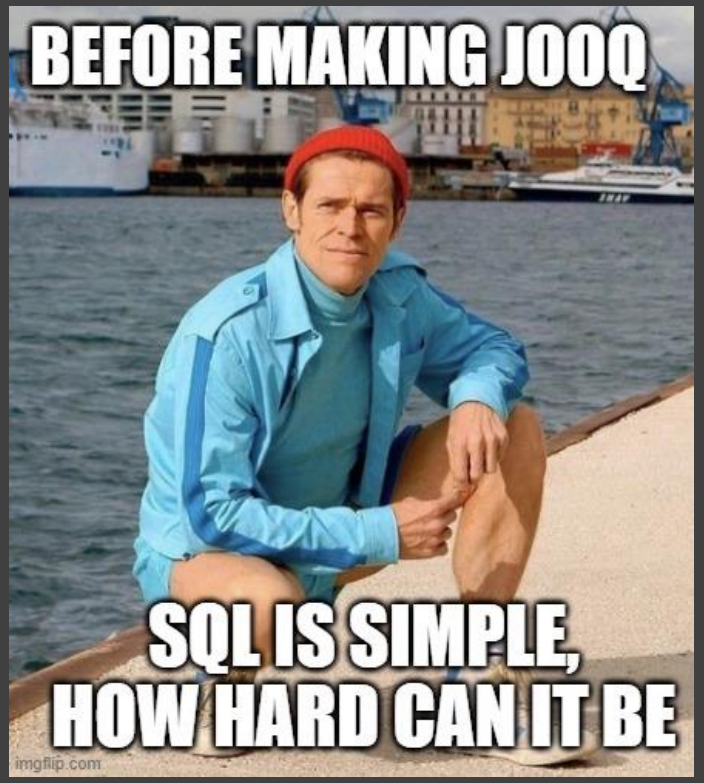
```
alter table events
  add constraint events_account_fk
  foreign key(account_id) references accounts(id)
  not valid
```

https://xata.io/blog/going-down-the-rabbit-hole-of-postgres-18-features

Herve Schweitzer (https://www.doag.org/formes/pubfiles/12736375/docs/Publikationen/DOAGNews/2021/01-2021/Seiten-aus-01_2021-Red_Stack_Magazin-Herve_Schweitzer-Wie_PostgreSQL_von_20_Jahren_Oracle_Tuning_profitieren_kann.pdf)

Fazit

Fazit


BEFORE MAKING JOOQ
SQL IS SIMPLE,
HOW HARD CAN IT BE
imgflip.com

Fazit


BEFORE MAKING JOOQ
SQL IS SIMPLE,
HOW HARD CAN IT BE
imgflip.com
13 YEARS LATER
THIS RDBMS DOES
WHAT WITH *WHAT SYNTAX*??

Fazit

- Wir haben im SQL und in der Architektur Unterschiede gesehen
- Wenn es keine guten Gründe gibt, starte **neue** Projekte mit PostgreSQL
- Kenne beide Seiten und bleib’ in Übung

Empfehlungen

- Postgres Weekly postgres@cooperpress.com
- Franck Pachot
- Hironobu Suzuki (https://www.interdb.jp/pg/index.html)
- Frits Hoogland (https://dev.to/fritshooglandyugabyte)
- Markus Winand (https://use-the-index-luke.com/de)

Empfehlungen

Sign up für Mailinglisten

https://www.postgresql.org/list/

Empfehlungen

Sign up für Mailinglisten

https://www.postgresql.org/list/

```
>
> regards, tom lane
>
```