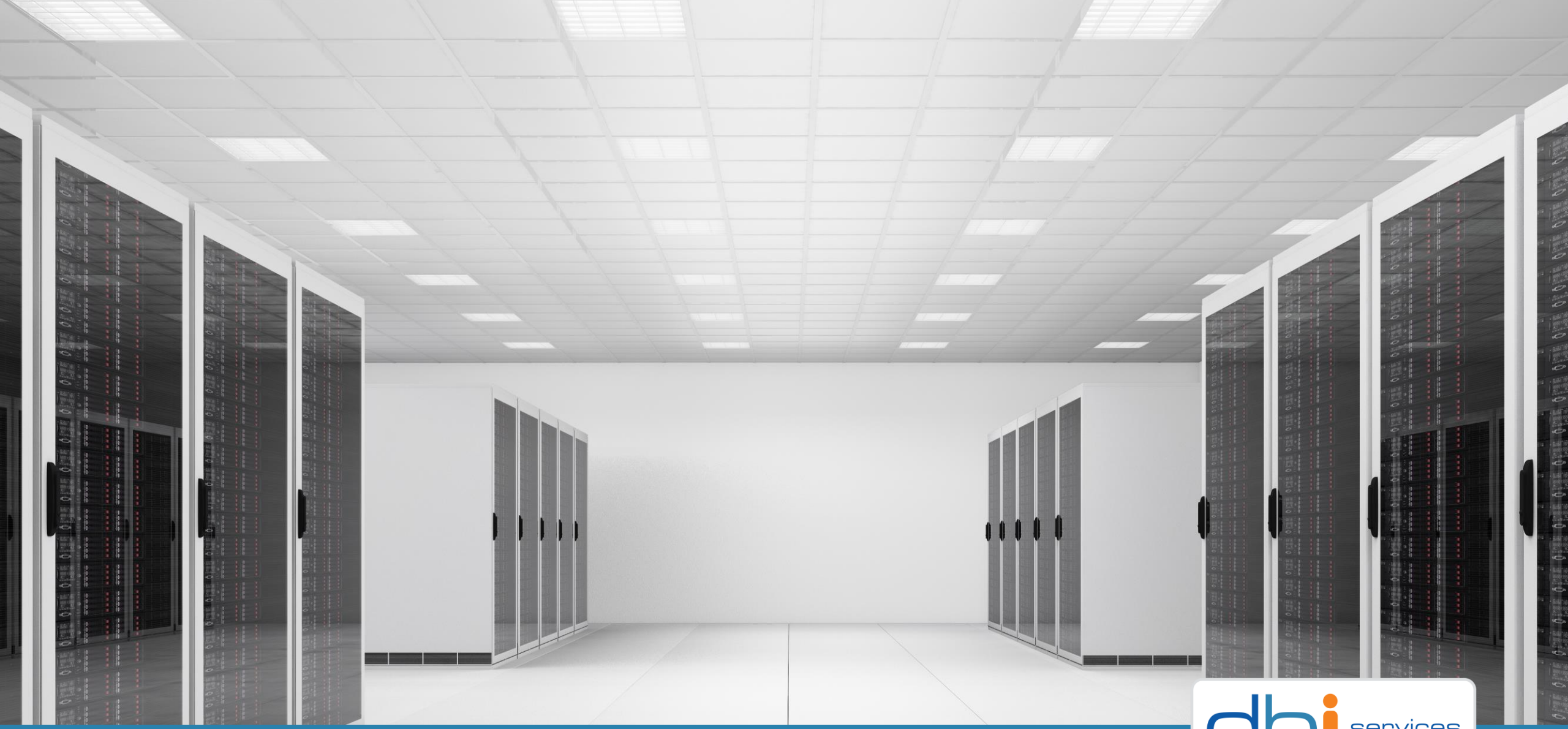


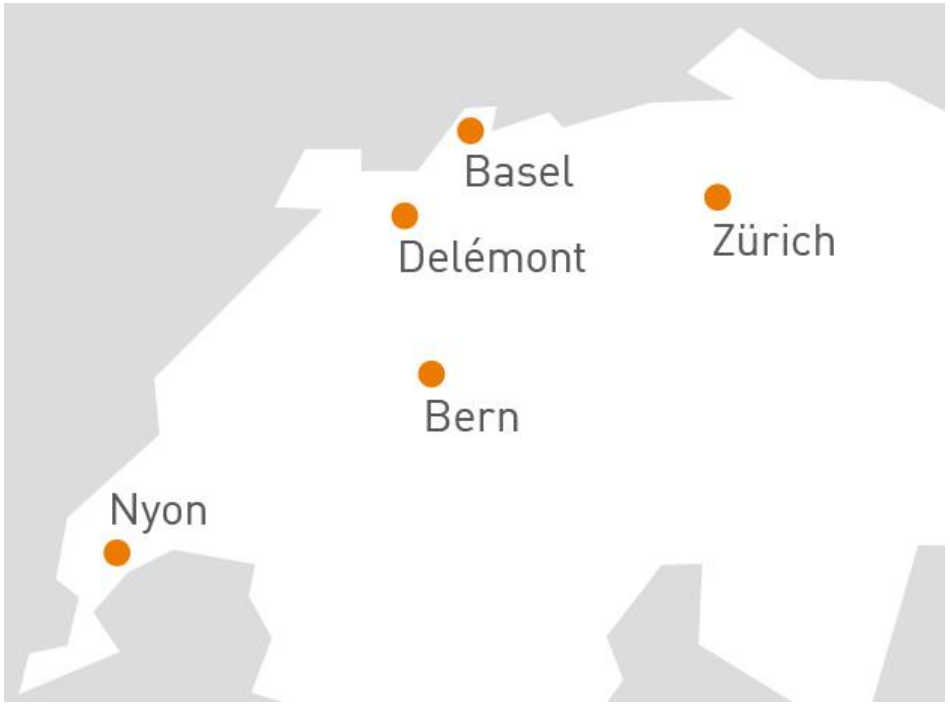
dbi services by Sequotech

Breaking PostgreSQL - Learning from doing it wrong

Who we are

The Company

- Founded in 2010
- More than 100 employees
- Specialized in the Middleware Infrastructure
  - The invisible part of IT
- Customers in Switzerland and all over Europe



Our Offer

- Consulting
- Service Level Agreements (SLA)
- Trainings
- License Management

About me

Daniel Westermann

Technology Leader

Principal Consultant

+41 79 927 24 46

daniel.westermann@dbi-services.com

Daniel Westermann

https://mastodon.social/@danielwestermann

Disclaimer

This is not so much about breaking PostgreSQL ...

... but more about doing it wrong ...

... and what you should avoid doing!


No agenda … Just demo ….

You'll find everything in the slides!

We're doing this on the development branch

The development branch gives you access to the latest commits

```
postgres=# select version();
                              version
-------------------------------------------------------------------
 PostgreSQL 18devel on x86_64-linux, compiled by gcc-14.2.1, 64-bit
(1 row)

postgres=# \! cd /home/postgres/postgresql; git log
commit 588acf6d0ec15d9384c2f712585c0f56936d7bac (HEAD -> master, origin/master, origin/HEAD)
Author: Amit Kapila <akapila@postgresql.org>
Date:   Thu Mar 6 14:19:38 2025 +0530

    Avoid invalidating all RelationSyncCache entries on publication change.
```

Here are the steps to get that up and running

Installing PostgreSQL from source code with Meson

```
postgres@pgbox:/home/postgres/$ sudo apt update -y && sudo apt dist-upgrade -y
postgres@pgbox:/home/postgres/$ sudo apt install libldap2-dev libpython3-dev libreadline-dev
libssl-dev bison flex libghc-zlib-dev libcrypto++-dev libxml2-dev libxslt1-dev tcl tclcl-dev
bzip2 wget screen libpam0g-dev libperl-dev make unzip libpam0g-dev python3 libsystemd-dev
sudo llvm llvm-dev clang pkg-config gcc g++ liblz4-dev pkg-config python3-distutils libbz2-
dev tuned libzstd-dev libossp-uuid-dev libkrb5-dev libyaml-dev python3-distro meson git
postgres@pgbox:/home/postgres/$ git clone https://git.postgresql.org/git/postgresql.git
postgres@pgbox:/home/postgres/$ mkdir build; cd $_
postgres@pgbox:/home/postgres/$ meson setup . ../postgresql
postgres@pgbox:/home/postgres/$ meson configure
postgres@pgbox:/home/postgres/$ ninja
postgres@pgbox:/home/postgres/$ sudo ninja install
postgres@pgbox:/home/postgres/$ cd; /usr/local/pgsql/bin/initdb --pgdata=~/dummy
postgres@pgbox:/home/postgres/$ /usr/local/pgsql/bin/pg_ctl --pgdata=~/dummy start
postgres@pgbox:/home/postgres/$ /usr/local/pgsql/bin/psql
psql (18devel)
Type "help" for help.

postgres=#
```

Let's start?

Don't use NOT IN with NULLS

Be careful with NULLs and NOT IN

```
postgres@pgbox:/home/postgres/$ psql -c "create table t ( a int );"
postgres@pgbox:/home/postgres/$ psql -c "insert into t values (1),(null);"
postgres@pgbox:/home/postgres/$ psql -c "select * from t where a not in (2,null);"
a
---
(0 rows)
postgres@pgbox:/home/postgres/$ psql -c "drop table t;"
```

Why?

> https://wiki.postgresql.org/wiki/Don't_Do_This

> This happens because col IN (1,null) returns TRUE if col=1,

> and NULL otherwise (i.e. it can never return FALSE).

> Since NOT (TRUE) is FALSE, but NOT (NULL) is still NULL,

> there is no way that NOT (col IN (1,null)) (which is the same thing as col NOT IN (1,null))

> can return TRUE under any circumstances.

With great power, comes great responsibility

It is your responsibility to provide something which makes sense

```
postgres@pgbox:/home/postgres/$ psql -c "alter system set archive_command='pg_ctl restart'";
postgres@pgbox:/home/postgres/$ psql -c "select pg_reload_conf()";
postgres@pgbox:/home/postgres/$ psql -c "select pg_switch_wal()";

2025-03-07 09:28:56.854 CET - 3 - 1334 -  - @ - 0LOG:  shutting down
2025-03-07 09:29:57.100 CET - 1 - 1339 -  - @ - 0LOG:  archive command failed with exit code
1
2025-03-07 09:29:57.100 CET - 2 - 1339 -  - @ - 0DETAIL:  The failed archive command was:
pg_ctl restart
waiting for server to shut down.....2025-03-07 09:30:02.654 CET - 1 - 1898 - [local] -
postgres@postgres - 0FATAL:  the database system is shutting down

postgres@pgbox:/home/postgres/$ pg_ctl stop
postgres@pgbox:/home/postgres/$ echo "archive_command='/bin/true'" >> \
                                     $PGDATA/postgresql.auto.conf
postgres@pgbox:/home/postgres/$ pg_ctl start
```

LIMIT NULL

What does LIMIT NULL do?

```
postgres@pgbox:/home/postgres/$ psql -c "create table t ( a int );"
postgres@pgbox:/home/postgres/$ psql -c "insert into t
                                         select * from generate_series(1,1000000) i;"
postgres@pgbox:/home/postgres/$ psql -c "select * from t limit null;"
postgres@pgbox:/home/postgres/$ psql -c "drop table t;"
```

Disabling constraints

In Oracle constraints can be temporarily disabled, don't do this in PostgreSQL

```
postgres@pgbox:/home/postgres/$ psql -c "create table t ( a int unique );"
postgres@pgbox:/home/postgres/$ psql -c "create table tt ( a int references t(a));"
postgres@pgbox:/home/postgres/$ psql -c "insert into t values (1),(2),(3);"
postgres@pgbox:/home/postgres/$ psql -c "insert into t values (1),(2),(3);"
postgres@pgbox:/home/postgres/$ psql -c "insert into tt values (1),(2),(3);"
postgres@pgbox:/home/postgres/$ psql -c "alter table tt disable trigger all;"
postgres@pgbox:/home/postgres/$ psql -c "\d tt"
postgres@pgbox:/home/postgres/$ psql -c "insert into tt values(4);"
postgres@pgbox:/home/postgres/$ psql -c "alter table tt validate constraint tt_a_fkey;"
postgres@pgbox:/home/postgres/$ psql -c "select convalidated
                                           from pg_constraint
                                          where conname = 'tt_a_fkey';"
postgres@pgbox:/home/postgres/$ psql -c "alter table tt enable trigger all;"
postgres@pgbox:/home/postgres/$ psql -c "alter table tt validate constraint tt_a_fkey;"
postgres@pgbox:/home/postgres/$ psql -c "insert into tt values(5);"
postgres@pgbox:/home/postgres/$ psql -c "select * from tt;"
postgres@pgbox:/home/postgres/$ psql -c "drop table t,tt;"
```

> https://www.postgresql.org/message-id/CAAJ_b962c5AcYW9KUt_R_ER5qs3fUGbe4az-SP-vuwPS-w-AGA@mail.gmail.com

Don't mess with the file system and enable checksums

You want your data to be safe

```
postgres@pgbox:/home/postgres/$ psql -c "create table t ( a int );"
postgres@pgbox:/home/postgres/$ psql -c "insert into t
                                         select * from generate_series(1,1000000) i;"
postgres@pgbox:/home/postgres/$ psql -c "select pg_relation_filepath('t');"
postgres@pgbox:/home/postgres/$ dd if=/dev/zero of=$PGDATA/base/5/16419 bs=8kB count=10
postgres@pgbox:/home/postgres/$ psql -c "select count(*) from t;"
postgres@pgbox:/home/postgres/$ pg_ctl stop
postgres@pgbox:/home/postgres/$ pg_ctl start
postgres@pgbox:/home/postgres/$ psql -c "select count(*) from t;"
postgres@pgbox:/home/postgres/$ pg_controldata | grep checksum
```

Respect the 1GB limit

Don't misuse a relational database

```
postgres@pgbox:/home/postgres/$ dd if=/dev/zero of=stupid.file bs=1M count=1002
postgres@pgbox:/home/postgres/$ psql -c "create table t ( a bytea );"
postgres@pgbox:/home/postgres/$ psql -c "insert into t
                                  select pg_read_binary_file('/home/postgres/stupid.file');"
postgres@pgbox:/home/postgres/$ psql -c "select * from t;"
postgres@pgbox:/home/postgres/$ pg_dump > a.sql
postgres@pgbox:/home/postgres/$ psql -c "drop table t;"
```

> https://www.postgresql.org/docs/current/limits.html

Be aware of what is logged

Logs can contain sensitive information

```
postgres@pgbox:/home/postgres/$ psql -c "alter user postgres
                                         with password 'not_so_secure';"; taa
postgres@pgbox:/home/postgres/$ psql -c "select pg_current_logfile()"
pg_current_logfile
---------------------------
 pg_log/postgresql-Fri.log
(1 row)
postgres@pgbox:/home/postgres/$ tail -20f $PGDATA/pg_log/postgresql-Fri.log

2025-03-07 09:50:31.642 CET - 1 - 2128 - [local] - postgres@postgres - 0LOG:  statement:
alter user postgres with password 'not_so_secure';

postgres@pgbox:/home/postgres/$ psql -c "\password"
postgres@pgbox:/home/postgres/$ tail -20f $PGDATA/pg_log/postgresql-Fri.log

2025-03-07 09:59:32.801 CET - 1 - 2193 - [local] - postgres@postgres - 0LOG:  statement:
ALTER USER "postgres" PASSWORD 'SCRAM-SHA-
256$4096:US/mEC1TaREedKXPcCGT6A==$3Fggx/g6FV5Ds3CiP0hanqfgg42b1TWKGC0NPU46LgE=:BJHf3PRJj/cTU
70nM0J8fALkBwKb81QRvVRdb/Lp808='
```

Don't mess with the catalog

Only because you can doesn't mean you should do something

```
postgres@pgbox:/home/postgres/$ psql -c "alter system set allow_system_table_mods = true;"
postgres@pgbox:/home/postgres/$ psql -c "select pg_reload_conf()";
postgres@pgbox:/home/postgres/$ psql -c "create table t ( a int );"
postgres@pgbox:/home/postgres/$ psql -c "update pg_class
                                            set reloptions = array['do not do this‟']
                                          where relname ='t';"
postgres@pgbox:/home/postgres/$ psql -c "alter table t set ( fillfactor = 99 );"

ERROR:  unrecognized parameter "do not do this‟"

postgres@pgbox:/home/postgres/$ psql -c "drop table t;"
```

This is not meant for end users! Even if you read it on the internet!

Don't be stupid

Be careful with recursion

```
postgres@pgbox:/home/postgres/$ psql -c 'create function f() returns void
                                         as $$ select f(); $$ language sql;'
postgres@pgbox:/home/postgres/$ clear;
postgres@pgbox:/home/postgres/$ psql -c "select f();" > x 2>&1
postgres@pgbox:/home/postgres/$ head x

ERROR:  stack depth limit exceeded
HINT:  Increase the configuration parameter "max_stack_depth" (currently 2048kB), after
ensuring the platform's stack depth limit is adequate.
CONTEXT:  SQL function "f" during inlining


postgres@pgbox:/home/postgres/$ psql -c "drop function f();"
```

Recursion to death!

Be nice to your colleagues

A blank is a valid character

```
postgres@pgbox:/home/postgres/$ psql -c "create database \"    \""
postgres@pgbox:/home/postgres/$ psql -c "create schema \"    \"" "    "
postgres@pgbox:/home/postgres/$ psql -c "create table \"    \".\"    \"(a int);" "    "
postgres@pgbox:/home/postgres/$ psql -l
postgres@pgbox:/home/postgres/$ psql -c "\dn" "    "
postgres@pgbox:/home/postgres/$ psql -c "\d \"    \".*" "    "
```

Of course you can, but should you?

Be even nicer to your colleagues

This is just for fun

```
postgres@pgbox:/home/postgres/$ psql -c "alter database template1 is_template = false;"
postgres@pgbox:/home/postgres/$ psql -c "drop database template1;"
postgres@pgbox:/home/postgres/$ psql -c "create database d;"

ERROR:  template database "template1" does not exist

postgres@pgbox:/home/postgres/$ psql -c "alter database template0
                                         is_template = false;" "    "
postgres@pgbox:/home/postgres/$ psql -c "drop database template0" "    "
postgres@pgbox:/home/postgres/$ psql -c "drop database postgres" "    "
postgres@pgbox:/home/postgres/$ psql -l
postgres@pgbox:/home/postgres/$ psql -l "    "
postgres@pgbox:/home/postgres/$ psql -c "create database template0
                                         with template=\"    \"" "    "
postgres@pgbox:/home/postgres/$ psql -c "create database template1
                                         with template=\"    \"" "    "
postgres@pgbox:/home/postgres/$ psql -c "create database postgres" "    "
postgres@pgbox:/home/postgres/$ psql -l
```

Don't turn off autovacuum

No, no, please don't do it

```
postgres@pgbox:/home/postgres/$ ps aux | grep autov
postgres@pgbox:/home/postgres/$ psql -c "alter system set autovacuum=off"
postgres@pgbox:/home/postgres/$ psql -c "select pg_reload_conf()"
postgres@pgbox:/home/postgres/$ ps aux | grep autov
postgres@pgbox:/home/postgres/$ psql -c "create table t ( a int, b text, c timestamptz )"
postgres@pgbox:/home/postgres/$ psql -c "insert into t select i, i::text, now()
                                         from generate_series(1,1000000) i"
postgres@pgbox:/home/postgres/$ psql -c "select pg_size_pretty ( pg_relation_size ('t') )"
postgres@pgbox:/home/postgres/$ psql -c "update t set b = 'aaa'"
postgres@pgbox:/home/postgres/$ psql -c "select pg_size_pretty ( pg_relation_size ('t') )"
postgres@pgbox:/home/postgres/$ psql -c "delete from t"
postgres@pgbox:/home/postgres/$ psql -c "select pg_size_pretty ( pg_relation_size ('t') )"
postgres@pgbox:/home/postgres/$ psql -c "vacuum"
postgres@pgbox:/home/postgres/$ psql -c "drop table t"
postgres@pgbox:/home/postgres/$ psql -c "select pg_size_pretty ( pg_relation_size ('t') )"
postgres@pgbox:/home/postgres/$ psql -c "alter system set autovacuum=on"
postgres@pgbox:/home/postgres/$ psql -c "select pg_reload_conf()"
postgres@pgbox:/home/postgres/$ ps aux | grep autov
```

… even when you read it on the internet!

A type is a type is a table is a type

Everything is a type and can be used with other types

```
postgres@pgbox:/home/postgres/$ psql -c "create table t ( a int )"
postgres@pgbox:/home/postgres/$ psql -c "create table tt ( a t )"
postgres@pgbox:/home/postgres/$ psql -c "create table ttt ( a t, b tt )"
postgres@pgbox:/home/postgres/$ psql -c "insert into t values (1);"
postgres@pgbox:/home/postgres/$ psql -c "insert into tt values (row(1));"
postgres@pgbox:/home/postgres/$ psql -c "insert into ttt values ( row(1), row(row(1)) );"
postgres@pgbox:/home/postgres/$ psql -c "select * from ttt;"
postgres@pgbox:/home/postgres/$ psql -c "create table tttt ( a tt[], b ttt[] )"
postgres@pgbox:/home/postgres/$ psql -c "drop table t,tt,ttt,tttt"
```

Don't make it too complex!

A NULL is a real NULL

NULL, NULL, NULL

```
postgres@pgbox:/home/postgres/$ psql -c "select (null is null) is true;"
postgres@pgbox:/home/postgres/$ psql -c "select (null = null) is true;"
postgres@pgbox:/home/postgres/$ psql -c "select ('' is null) is true;"
postgres@pgbox:/home/postgres/$ psql -c "show transform_null_equals"
```

NULL means undefined

19.13.2. Platform and Client Compatibility

`transform_null_equals` (`boolean`)

When on, expressions of the form ***expr*** `= NULL` (or `NULL =` ***expr***) are treated as ***expr*** `IS NULL`, that is, they return true if ***expr*** evaluates to the null value, and false otherwise. The correct SQL-spec-compliant behavior of ***expr*** `= NULL` is to always return null (unknown). Therefore this parameter defaults to `off`.

Orphaned files

When a transaction dies ...

```
postgres@pgbox:/home/postgres/$ psql <<< "begin;
select pg_backend_pid();
create table x ( a int );
select pg_relation_filepath('x');
select pg_sleep(120);"

postgres@pgbox:/home/postgres/$ kill -9 PID
postgres@pgbox:/home/postgres/$ ls -l $PGDATA/base/16430/16454
-rw-------. 1 postgres postgres 0 Mar  7 10:43 /u02/pgdata/PGDEV//base/16430/16454

postgres@pgbox:/home/postgres/$ pg_ctl restart
postgres@pgbox:/home/postgres/$ ls -l $PGDATA/base/16430/16454
-rw-------. 1 postgres postgres 0 Mar  7 10:43 /u02/pgdata/PGDEV//base/16430/16454
```

... you might have orphaned files on disk

Unlogged tables and physical backups

Know what you do, it most probably comes with consequences

```
postgres@pgbox:/home/postgres/$ psql -c "create unlogged table t ( a int )"
postgres@pgbox:/home/postgres/$ psql -c "insert into t
                                         select * from generate_series(1,1000)"
postgres@pgbox:/home/postgres/$ mkdir /var/tmp/dummy
postgres@pgbox:/home/postgres/$ pg_basebackup --pgdata=/var/tmp/dummy --checkpoint=fast
postgres@pgbox:/home/postgres/$ echo "port=8888" >> /var/tmp/dummy/postgresql.auto.conf
postgres@pgbox:/home/postgres/$ chmod 700 /var/tmp/dummy
postgres@pgbox:/home/postgres/$ pg_ctl --pgdata=/var/tmp/dummy start
postgres@pgbox:/home/postgres/$ psql -p 8888 -c "select count(*) from t"
postgres@pgbox:/home/postgres/$ pg_ctl --pgdata=/var/tmp/dummy stop -m immediate
postgres@pgbox:/home/postgres/$ rm -rf /var/tmp/dummy/*
postgres@pgbox:/home/postgres/$ pg_dump  > d
postgres@pgbox:/home/postgres/$ grep -A 5 COPY d
```

Unlogged tables do NOT go into a physical backup (there no WAL)

Unlogged tables and physical backups

Know what you do, it most probably comes with consequences

```
postgres@pgbox:/home/postgres/$ psql -c "alter table t set logged"
postgres@pgbox:/home/postgres/$ psql -c "insert into t
                                         select * from generate_series(1,1000)"
postgres@pgbox:/home/postgres/$ psql -c "select count(*) from t"
postgres@pgbox:/home/postgres/$ pg_basebackup --pgdata=/var/tmp/dummy --checkpoint=fast
postgres@pgbox:/home/postgres/$ echo "port=8888" >> /var/tmp/dummy/postgresql.auto.conf
postgres@pgbox:/home/postgres/$ chmod 700 /var/tmp/dummy
postgres@pgbox:/home/postgres/$ pg_ctl --pgdata=/var/tmp/dummy start
postgres@pgbox:/home/postgres/$ psql -p 8888 -c "select count(*) from t"
postgres@pgbox:/home/postgres/$ pg_ctl --pgdata=/var/tmp/dummy stop -m immediate
postgres@pgbox:/home/postgres/$ rm -rf /var/tmp/dummy/*
```

Deferrable constraints

Constraint validation

```
postgres@pgbox:/home/postgres/$ psql -c "create table x ( a int primary key)"
postgres@pgbox:/home/postgres/$ psql -c "insert into x values (1),(2)"
postgres@pgbox:/home/postgres/$ psql -c "update x set a = a + 1"
postgres@pgbox:/home/postgres/$ psql -c "create table xx ( a int primary key deferrable)"
postgres@pgbox:/home/postgres/$ psql -c "insert into xx values (1),(2)"
postgres@pgbox:/home/postgres/$ psql -c "update xx set a = a + 1"
postgres@pgbox:/home/postgres/$ psql -c "drop table x,xx"
```

In Oracle that works without "deferrable"

PL/pgSQL and recursion

Avoid many, many nested calls to functions and/or procedures

```
postgres@pgbox:/home/postgres/$ time psql -f plpgsql-recursion.sql

real    0m53.913s
user    0m0.005s
sys     0m0.007s

postgres@pgbox:/home/postgres/$ vi plpgsql-recursion.sql
```

https://www.postgresql.org/message-id/flat/ZR0P278MB0920DA81D97D5F30D4A46243D2EC9%40ZR0P278MB0920.CHEP278.PROD.OUTLOOK.COM

Login event triggers

Make sure, your login event triggers are safe

```
postgres@pgbox:/home/postgres/$ cat trigger.sql
create or replace function deny_session_daniel()
  returns event_trigger security definer
  as
$$
  declare
  begin
      raise exception 'user %1 is not allowed to login', session_user;
  end;
$$ language plpgsql;

create event trigger my_login_trg
  on login
  execute function deny_session_daniel();

alter event trigger my_login_trg enable always;
```

Login event triggers

Make sure, your login event triggers are safe

```
postgres@pgbox:/home/postgres/$ psql -f trigger.sql
CREATE FUNCTION
CREATE EVENT TRIGGER
ALTER EVENT TRIGGER

postgres@pgbox:/home/postgres/$ psql
psql: error: connection to server on socket "/tmp/.s.PGSQL.5432" failed: FATAL:  user
postgres1 is not allowed to login
CONTEXT:  PL/pgSQL function deny_session_daniel() line 4 at RAISE
```

Login event triggers

Make sure, your login event triggers are safe

```
postgres@pgbox:/home/postgres/$ psql -c "alter system set event_triggers to off" template1
postgres@pgbox:/home/postgres/$ psql -c "select pg_reload_conf()" template1
postgres@pgbox:/home/postgres/$ psql
postgres@pgbox:/home/postgres/$ psql -c "drop event trigger my_login_trg"
```

COPY FROM PROGRAM

Again, with great power comes great responsibility

```
postgres@pgbox:/home/postgres/$ psql -c "create table s ( a text )"
postgres@pgbox:/home/postgres/$ psql -c "copy s from program 'cat /etc/passwd'"
postgres@pgbox:/home/postgres/$ psql -c "select * from s"
postgres@pgbox:/home/postgres/$ psql -c "copy s from program
                                         'initdb --pgdata=/var/tmp/dummy'"
postgres@pgbox:/home/postgres/$ psql -c "copy s from program
                                         'cat /home/postgres/.ssh/id_ed25519'"
postgres@pgbox:/home/postgres/$ psql -c "select * from s"
```

This is restricted to Superusers by default!

Multi column indexes

Not the same behavior than Oracle

```
postgres@pgbox:/home/postgres/$ psql -c "create table k ( a int, b int )"
postgres@pgbox:/home/postgres/$ psql -c "insert into k
                                         select i,i+1 from generate_series(1,1000000) i"
postgres@pgbox:/home/postgres/$ psql -c "create index i on k (a,b)"
postgres@pgbox:/home/postgres/$ psql -c "explain select * from k where a = 1"
postgres@pgbox:/home/postgres/$ psql -c "explain select * from k where b = 2"
postgres@pgbox:/home/postgres/$ psql -c "create index i2 on k(b)"
postgres@pgbox:/home/postgres/$ psql -c "explain select * from k where b = 2"
```

It is usually better to create multiple indexes. There is a patch to improve this:

https://www.postgresql.org/message-id/CAH2-Wzmn1YsLzOGgjAQZdn1STSG_y8qP__vggTaPAYXJP+G4bw@mail.gmail.com

Trust authentication for local connections

"trust" is _not_ safe, even for socket connections

```
postgres@pgbox:/home/postgres/$ cat $PGDATA/pg_hba.conf | grep local | grep -v "^#"
postgres@pgbox:/home/postgres/$ sudo useradd u
postgres@pgbox:/home/postgres/$ sudo su - u
postgres@pgbox:/home/postgres/$ /u01/app/postgres/product/DEV/db_0/bin/psql postgres
```

No, please don't

Don't use the CHAR data type

There is no use case for CHAR

```
postgres@pgbox:/home/postgres/$ psql -c "create table p ( a char(100) )"
postgres@pgbox:/home/postgres/$ psql -c "create table pp ( a varchar(100) )"
postgres@pgbox:/home/postgres/$ psql -c "insert into p select 'a'
                                         from generate_series(1,1000000)"
postgres@pgbox:/home/postgres/$ psql -c "insert into pp select 'a'
                                         from generate_series(1,1000000)"
postgres@pgbox:/home/postgres/$ psql -c "select pg_relation_size('p')"
postgres@pgbox:/home/postgres/$ psql -c "select pg_relation_size('pp')"
```

CHAR is blank padded

Don't use SERIALS for primary keys

This is not want you want

```
postgres@pgbox:/home/postgres/$ psql -c "create sequence s1"
postgres@pgbox:/home/postgres/$ psql -c "create table a ( a int primary key
                                        default nextval('s1'), b int)"
postgres@pgbox:/home/postgres/$ psql -c "insert into a (b) values (1)"
postgres@pgbox:/home/postgres/$ psql -c "insert into a (a,b) values (2,2)"
postgres@pgbox:/home/postgres/$ psql -c "select * from a"
postgres@pgbox:/home/postgres/$ psql -c "insert into a(b) values (3)"
postgres@pgbox:/home/postgres/$ psql -c "create table aa ( a int primary key
                                        generated always as identity, b int)"
postgres@pgbox:/home/postgres/$ psql -c "insert into aa values (1,1)"
```

> Don't use SERIAL at all

Between and ">= and <" are not the same thing

Be careful with between

```
postgres@pgbox:/home/postgres/$ psql -c "drop table t;"
postgres@pgbox:/home/postgres/$ psql -c "create table t ( a timestamp );"
postgres@pgbox:/home/postgres/$ psql -c "insert into t values ('2025-03-26');"
postgres@pgbox:/home/postgres/$ psql -c "insert into t values ('2025-03-27');"
postgres@pgbox:/home/postgres/$ psql -c "insert into t values ('2025-03-28');"
postgres@pgbox:/home/postgres/$ psql -c "insert into t values ('2025-03-28');"
postgres@pgbox:/home/postgres/$ psql -c "select * from t;"
postgres@pgbox:/home/postgres/$ psql -c "select * from t
                                         where a between '2025-03-26' and '2025-03-28';"
postgres@pgbox:/home/postgres/$ psql -c "select * from t
                                         where a >= '2025-03-26' and a <  '2025-03-28';"
```

Check constraints and functions

Changing a function does not re-execute constraint validation

```
postgres@pgbox:/home/postgres/$ psql -c 'create function f1(int) returns boolean as $$
                                         select $1 > 1 $$ language sql;';
postgres@pgbox:/home/postgres/$ psql -c "create table r ( a int check (f1(a)));"
postgres@pgbox:/home/postgres/$ psql -c "insert into r values(0);"
postgres@pgbox:/home/postgres/$ psql -c "insert into r values(2);"
postgres@pgbox:/home/postgres/$ psql -c 'create or replace function f1(int)
                                        returns boolean as $$ select $1 < 1 $$ language sql;'
postgres@pgbox:/home/postgres/$ psql -c "select * from r"
```

Check constraints and NULLs

Whatch out for generic plans

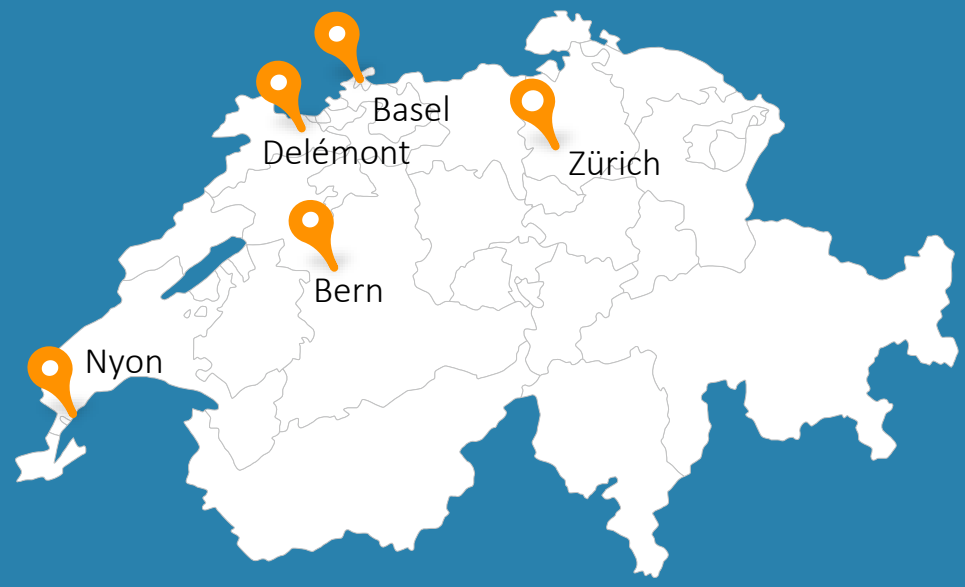
```
postgres@pgbox:/home/postgres/$ psql -c "drop table t"
postgres@pgbox:/home/postgres/$ psql -c "create table t ( a int );"
postgres@pgbox:/home/postgres/$ psql -c "create index i on t(a);"
postgres@pgbox:/home/postgres/$ psql -c "insert into t select 1 from
                                         generate_series(1,1000000);"
postgres@pgbox:/home/postgres/$ psql -c "insert into t values
                                          (2),(3),(4),(5),(6),(7),(8),(9);"
postgres@pgbox:/home/postgres/$ psql -c "analyze t;"
postgres@pgbox:/home/postgres/$ psql <<< '
prepare my_stmt as select * from t where a = $1;
explain execute my_stmt(1);
explain execute my_stmt(1);
explain execute my_stmt(1);
explain execute my_stmt(1);
explain execute my_stmt(1);
explain execute my_stmt(1);
explain execute my_stmt(1);
explain execute my_stmt(2);
'
postgres@pgbox:/home/postgres/$ psql -c "explain select * from t where a = 2;"
postgres@pgbox:/home/postgres/$ psql -c "drop table t"
```

Do you want to play a PostgreSQL game?

https://github.com/danielwestermann/ThePostgreSQLGame

Any questions?

Please do ask!



We would love to boost your IT-Infrastructure

How about you?