### 

### psql -p 5432 -U postgres -d postgres

### psql: could not connect to server: No such file or directory

### Is the server running locally and accepting

### connections on Unix domain socket "/tmp/.s.PGSQL.5432"?

$ **pg\_ctl -D $PGDATA status**

pg\_ctl: no server running

$ **pg\_ctl -D $PGDATA start**

waiting for server to start....2019-09-05 22:47:51.551 IST [87688] LOG: listening on IPv6 address "::1", port 5432

2019-09-05 22:47:51.552 IST [87688] LOG: listening on IPv4 address "127.0.0.1", port 5432

2019-09-05 22:47:51.555 IST [87688] LOG: listening on Unix socket "/tmp/.s.PGSQL.5432"

2019-09-05 22:47:51.576 IST [87689] LOG: database system was shut down at 2019-09-05 22:46:37 IST

2019-09-05 22:47:51.621 IST [87688] LOG: database system is ready to accept connections

done

server started

$ **pg\_ctl -D $PGDATA status**

pg\_ctl: server is running (PID: 87688)

/Users//pg\_software/11.5/bin/postgres "-D" "/Users//pg\_software/11.5/data"

An alternate option to check whether PostgreSQL is running is to check the process status as follows

$ **ps -ef|grep postgres**

1363659639 87690 87688 0 10:47PM ?? 0:00.00 postgres: checkpointer

1363659639 87691 87688 0 10:47PM ?? 0:00.01 postgres: background writer

1363659639 87692 87688 0 10:47PM ?? 0:00.05 postgres: walwriter

1363659639 87693 87688 0 10:47PM ?? 0:00.05 postgres: autovacuum launcher

1363659639 87694 87688 0 10:47PM ?? 0:00.04 postgres: stats collector

1363659639 87695 87688 0 10:47PM ?? 0:00.05 postgres: logical replication launcher

1363659639 87688 1 0 10:47PM ttys002 0:00.03 /Users//pg\_software/11.5/bin/postgres -D /Users//pg\_software/11.5/data

1363659639 87705 27416 0 10:48PM ttys002 0:00.01 grep postgres

$

$ **grep -i port $PGDATA/postgresql.conf**

port = 5432 # (change requires restart)

psql -p 5435 -U postgres -h 192.168.225.185 postgres

psql: could not connect to server: Connection refused

Is the server running on host "192.168.225.185" and accepting

TCP/IP connections on port 5432?

### 

$ psql -p 5432 -U postgres -d postgres

psql (11.5)

Type "help" for help.

postgres=# show listen\_addresses ;

listen\_addresses

------------------

localhost

(1 row)

postgres=#

psql -p 5435 -U postgres -h 192.168.225.185 postgres

psql: FATAL: no pg\_hba.conf entry for host "192.168.225.130", user "postgres", database "postgres", SSL off

$ vi $PGDATA/pg\_hba.conf

host all all 192.168.225.130/32 trust

$ pg\_ctl -D $PGDATA reload

server signaled

$ psql -p 5435 -U postgres -h 192.168.225.185 postgres

psql (11.5)

Type "help" for help.

postgres=#

$ psql -p 5432 -U postgres -d postgres

psql.bin: FATAL: sorry, too many clients already

postgres=# show max\_connections ;

max\_connections

-----------------

2

(1 row)

postgres=# select count(\*) from pg\_stat\_activity;

count

-------

2

(1 row)

postgres=# select pid,query,state from pg\_stat\_activity where state like 'idle';

pid | query | state

-------+-------+-------

11855 | | idle

(1 row)

postgres=# select pg\_terminate\_backend(pid) from pg\_stat\_activity where state='idle' and pid <> pg\_backend\_pid();

pg\_terminate\_backend

----------------------

t

(1 row)

postgres=# select pid,query,state from pg\_stat\_activity where state like 'idle';

pid | query | state

-----+-------+-------

(0 rows)

psql -p 5432 -U test postgres

psql.bin: FATAL: remaining connection slots are reserved for non-replication superuser connections

ERROR: canceling statement due to statement timeout

:

hannu=# set statement\_timeout = '3 s';

SET

hannu=# select pg\_sleep (10);

ERROR: canceling statement due to statement timeout

postgres=# select \* from test;

ERROR: relation "test" does not exist

LINE 1: select \* from test;

^

postgres=# select quote\_literal(relname) from pg\_class where upper(relname)='TEST';

quote\_literal

---------------

'TesT'

(1 row)

postgres=# select \* from "TesT";

t

---

(0 rows)

postgres=# \d '\*'."TesT"

Table "test.TesT"

Column | Type | Modifiers

--------+---------+-----------

t | integer |

postgres=# select \* from "test"."TesT";

t

---

(0 rows)

postgres=# set search\_path to "test";

SET

postgres=# select \* from "TesT";

t

---

(0 rows)

testdb=# drop user bob;

ERROR: role "bob" cannot be dropped because some objects depend on it

DETAIL: owner of table bobstable

owner of sequence bobstable\_id\_seq

REASSIGN OWNED BY old\_role to new\_role;

DROP USER old\_role;

DROP OWNED BY name [, ...] [ CASCADE | RESTRICT ];

DROP user username;

"LOG: out of file descriptors: Too many open files in system; release and retry"

postgres=> copy test from '/tmp/test.txt';

ERROR: must be superuser to COPY to or from a file

HINT: Anyone can COPY to stdout or from stdin. psql's \copy command also works for anyone.

postgres=> select current\_user;

current\_user

--------------

test

(1 row)

postgres=> \copy test from '/tmp/test.txt';

postgres=> select \* from test;

t

---

1

2

3

4

5

(5 rows)

create or replace function copy\_for\_testuser(tablename text, filepath text)

returns void

security definer

as

$$

declare

begin

execute 'copy ' || tablename || ' from ''' || filepath || '''';

end;

$$ language plpgsql;

postgres=# \c postgres test

You are now connected to database "postgres" as user "test".

postgres=>

postgres=> select copy\_for\_testuser('test','/tmp/test.txt');

copy\_for\_testuser

-------------------

(1 row)

postgres=> select \* from test;

t

---

1

2

3

4

5

(5 rows)

ERROR: tablespace "old\_tablespace" is not empty

postgres=# INSERT INTO cust\_view

postgres-# VALUES (5, 'firstname', 'lastname', 133);

ERROR: cannot insert into a view

HINT: You need an unconditional ON INSERT DO INSTEAD rule.

CREATE RULE cust\_view\_insert AS

ON insert TO cust\_view

DO INSTEAD

INSERT INTO cust

VALUES (new.customerid, new.firstname, new.lastname, new.age);

And now retry our INSERT as follows:

postgres=# INSERT INTO cust\_view

postgres-# VALUES (5, 'firstname', 'lastname', 133);

INSERT 0 1

postgres=> set log\_min\_duration\_statement to 0;

ERROR: permission denied to set parameter "log\_min\_duration\_statement"

create or replace function debugging\_info\_on()

returns void

security definer

as

$$ begin

set client\_min\_messages to 'DEBUG1';

set log\_min\_messages to 'DEBUG1';

set log\_error\_verbosity to 'VERBOSE';

set log\_min\_duration\_statement to 0;

end;

$$ language plpgsql;

revoke all on function debugging\_info\_on() from public;

grant execute on function debugging\_info\_on() to bob;

postgres=# CREATE OR REPLACE VIEW test\_view

AS SELECT id as title2 FROM test;

ERROR: cannot change name of view column "title1" to "title2"

postgres=# CREATE OR REPLACE VIEW test\_view AS

postgres-# SELECT id as title1 FROM test;

CREATE VIEW

postgres=# CREATE OR REPLACE VIEW test\_view AS

postgres-# SELECT id as title2 FROM test;

ERROR: cannot change name of view column "title1" to "title2"

postgres=# DROP VIEW test\_view;

DROP VIEW

postgres=# CREATE OR REPLACE VIEW test\_view AS

postgres-# SELECT id as title2 FROM test;

CREATE VIEW

FATAL: could not create shared memory segment: Invalid argument

DETAIL: Failed system call was shmget(key=5440001, size=4011376640, 03600

In pg\_log, "pgstat wait timeout"

select version();

SELECT pg\_size\_pretty(pg\_database\_size('mydatabasename')) As fulldbsize;

\dt pg\_catalog.\*

select schemaname as "Schema Name", relname as "Table Name",

n\_tup\_ins+n\_tup\_upd+n\_tup\_del as "no.of writes" from

pg\_stat\_all\_tables where schemaname not in ('snapshots','pg\_catalog')

order by n\_tup\_ins+n\_tup\_upd+n\_tup\_del desc limit 10;

SELECT schemaname as "Schema Name", relname as "Table

Name",seq\_tup\_read+idx\_tup\_fetch as "no. of reads" FROM

pg\_stat\_all\_tables WHERE (seq\_tup\_read + idx\_tup\_fetch) > 0 and

schemaname NOT IN ('snapshots','pg\_catalog') ORDER BY

seq\_tup\_read+idx\_tup\_fetch desc limit 10;

SELECT QUOTE\_IDENT(TABLE\_SCHEMA)||'.'||QUOTE\_IDENT(table\_name) as

table\_name,pg\_relation\_size(QUOTE\_IDENT(TABLE\_SCHEMA)||'.'||QUOTE\_IDENT(table\_name)) as size,

pg\_total\_relation\_size(QUOTE\_IDENT(TABLE\_SCHEMA)||'.'||QUOTE\_IDENT(table\_name)) as total\_size,

pg\_size\_pretty(pg\_relation\_size(QUOTE\_IDENT(TABLE\_SCHEMA)||'.'||QUOTE\_IDENT(table\_name))) as pretty\_relation\_size,pg\_size\_pretty(pg\_total\_relation\_size(QUOTE\_IDENT(TABLE\_SCHEMA)||'.'||QUOTE\_IDENT(table\_name))) as pretty\_total\_relation\_size FROM information\_schema.tables WHERE QUOTE\_IDENT(TABLE\_SCHEMA) NOT IN ('snapshots') ORDER BY size DESC LIMIT 10;

SELECT datname, pg\_database\_size(datname),

pg\_size\_pretty(pg\_database\_size(datname))

FROM pg\_database

ORDER BY 2 DESC;

SELECT schemaname, relname, pg\_total\_relation\_size(schemaname

|| '.' || relname ) ,

pg\_size\_pretty(pg\_total\_relation\_size(schemaname || '.' ||

relname ))

FROM pg\_stat\_user\_tables

ORDER BY 3 DESC;

SELECT schemaname, relname, indexrelname,

pg\_total\_relation\_size(schemaname || '.' || indexrelname ) ,

pg\_size\_pretty(pg\_total\_relation\_size(schemaname || '.' ||

indexrelname ))

FROM pg\_stat\_user\_indexes

ORDER BY 1,2,3,4 DESC;

SELECT schemaname, relname, indexrelname, idx\_scan, idx\_tup\_fetch,

idx\_tup\_read

FROM pg\_stat\_user\_indexes

ORDER BY 4 DESC,1,2,3;

select relname, /\* pg\_size\_pretty( pg\_relation\_size( relid ) ) as table\_size,

pg\_size\_pretty( pg\_total\_relation\_size( relid ) ) as table\_total\_size, \*/

n\_tup\_upd, n\_tup\_hot\_upd, n\_live\_tup, n\_dead\_tup, last\_vacuum::date, last\_autovacuum::date, last\_analyze::date, last\_autoanalyze::date

from pg\_stat\_all\_tables

where relid in (select oid from pg\_class

where relnamespace not in (select oid from pg\_namespace

where nspname in ('information\_schema', 'pg\_catalog','pg\_toast', 'edbhc' ) ) )

order by n\_tup\_upd desc, schemaname, relname;

SELECT schemaname,

relname,

now() - last\_autovacuum AS "noautovac",

now() - last\_vacuum AS "novac",

n\_tup\_upd,

n\_tup\_del,

autovacuum\_count,

last\_autovacuum,

vacuum\_count,

last\_vacuum

FROM pg\_stat\_user\_tables

WHERE (now() - last\_autovacuum > '7 days'::interval

AND now() - last\_vacuum >'7 days'::interval)

OR (last\_autovacuum IS NULL AND last\_vacuum IS NULL ) AND n\_dead\_tup > 0

ORDER BY novac DESC;

SELECT relname, n\_live\_tup, n\_dead\_tup, trunc(100\*n\_dead\_tup/(n\_live\_tup+1))::float "ratio%",

to\_char(last\_autovacuum, 'YYYY-MM-DD HH24:MI:SS') as autovacuum\_date,

to\_char(last\_autoanalyze, 'YYYY-MM-DD HH24:MI:SS') as autoanalyze\_date

FROM pg\_stat\_all\_tables where schemaname not in ('pg\_toast','pg\_catalog','information\_schema')

ORDER BY last\_autovacuum;

SELECT current\_database(), nspname AS schemaname, tblname, idxname, bs\*(relpages)::bigint AS real\_size,

bs\*(relpages-est\_pages)::bigint AS extra\_size,

100 \* (relpages-est\_pages)::float / relpages AS extra\_ratio,

fillfactor, bs\*(relpages-est\_pages\_ff) AS bloat\_size,

100 \* (relpages-est\_pages\_ff)::float / relpages AS bloat\_ratio,

is\_na

-- , 100-(sub.pst).avg\_leaf\_density, est\_pages, index\_tuple\_hdr\_bm, maxalign, pagehdr, nulldatawidth, nulldatahdrwidth, sub.reltuples, sub.relpages -- (DEBUG INFO)

FROM (

SELECT coalesce(1 +

ceil(reltuples/floor((bs-pageopqdata-pagehdr)/(4+nulldatahdrwidth)::float)), 0 -- ItemIdData size + computed avg size of a tuple (nulldatahdrwidth)

) AS est\_pages,

coalesce(1 +

ceil(reltuples/floor((bs-pageopqdata-pagehdr)\*fillfactor/(100\*(4+nulldatahdrwidth)::float))), 0

) AS est\_pages\_ff,

bs, nspname, table\_oid, tblname, idxname, relpages, fillfactor, is\_na

-- , stattuple.pgstatindex(quote\_ident(nspname)||'.'||quote\_ident(idxname)) AS pst, index\_tuple\_hdr\_bm, maxalign, pagehdr, nulldatawidth, nulldatahdrwidth, reltuples -- (DEBUG INFO)

FROM (

SELECT maxalign, bs, nspname, tblname, idxname, reltuples, relpages, relam, table\_oid, fillfactor,

( index\_tuple\_hdr\_bm +

maxalign - CASE -- Add padding to the index tuple header to align on MAXALIGN

WHEN index\_tuple\_hdr\_bm%maxalign = 0 THEN maxalign

ELSE index\_tuple\_hdr\_bm%maxalign

END

+ nulldatawidth + maxalign - CASE -- Add padding to the data to align on MAXALIGN

WHEN nulldatawidth = 0 THEN 0

WHEN nulldatawidth::integer%maxalign = 0 THEN maxalign

ELSE nulldatawidth::integer%maxalign

END

)::numeric AS nulldatahdrwidth, pagehdr, pageopqdata, is\_na

-- , index\_tuple\_hdr\_bm, nulldatawidth -- (DEBUG INFO)

FROM (

SELECT

i.nspname, i.tblname, i.idxname, i.reltuples, i.relpages, i.relam, a.attrelid AS table\_oid,

current\_setting('block\_size')::numeric AS bs, fillfactor,

CASE -- MAXALIGN: 4 on 32bits, 8 on 64bits (and mingw32 ?)

WHEN version() ~ 'mingw32' OR version() ~ '64-bit|x86\_64|ppc64|ia64|amd64' THEN 8

ELSE 4

END AS maxalign,

/\* per page header, fixed size: 20 for 7.X, 24 for others \*/

24 AS pagehdr,

/\* per page btree opaque data \*/

16 AS pageopqdata,

/\* per tuple header: add IndexAttributeBitMapData if some cols are null-able \*/

CASE WHEN max(coalesce(s.null\_frac,0)) = 0

THEN 2 -- IndexTupleData size

ELSE 2 + (( 32 + 8 - 1 ) / 8) -- IndexTupleData size + IndexAttributeBitMapData size ( max num filed per index + 8 - 1 /8)

END AS index\_tuple\_hdr\_bm,

/\* data len: we remove null values save space using it fractionnal part from stats \*/

sum( (1-coalesce(s.null\_frac, 0)) \* coalesce(s.avg\_width, 1024)) AS nulldatawidth,

max( CASE WHEN a.atttypid = 'pg\_catalog.name'::regtype THEN 1 ELSE 0 END ) > 0 AS is\_na

FROM pg\_attribute AS a

JOIN (

SELECT nspname, tbl.relname AS tblname, idx.relname AS idxname, idx.reltuples, idx.relpages, idx.relam,

indrelid, indexrelid, indkey::smallint[] AS attnum,

coalesce(substring(

array\_to\_string(idx.reloptions, ' ')

from 'fillfactor=([0-9]+)')::smallint, 90) AS fillfactor

FROM pg\_index

JOIN pg\_class idx ON idx.oid=pg\_index.indexrelid

JOIN pg\_class tbl ON tbl.oid=pg\_index.indrelid

JOIN pg\_namespace ON pg\_namespace.oid = idx.relnamespace

WHERE pg\_index.indisvalid AND tbl.relkind = 'r' AND idx.relpages > 0

) AS i ON a.attrelid = i.indexrelid

JOIN pg\_stats AS s ON s.schemaname = i.nspname

AND ((s.tablename = i.tblname AND s.attname = pg\_catalog.pg\_get\_indexdef(a.attrelid, a.attnum, TRUE)) -- stats from tbl

OR (s.tablename = i.idxname AND s.attname = a.attname))-- stats from functionnal cols

JOIN pg\_type AS t ON a.atttypid = t.oid

WHERE a.attnum > 0

GROUP BY 1, 2, 3, 4, 5, 6, 7, 8, 9

) AS s1

) AS s2

JOIN pg\_am am ON s2.relam = am.oid WHERE am.amname = 'btree'

) AS sub

-- WHERE NOT is\_na

ORDER BY 2,3,4;

SELECT current\_database(), schemaname, tblname, bs\*tblpages AS real\_size,

(tblpages-est\_tblpages)\*bs AS extra\_size,

CASE WHEN tblpages - est\_tblpages > 0

THEN 100 \* (tblpages - est\_tblpages)/tblpages::float

ELSE 0

END AS extra\_ratio, fillfactor, (tblpages-est\_tblpages\_ff)\*bs AS bloat\_size,

CASE WHEN tblpages - est\_tblpages\_ff > 0

THEN 100 \* (tblpages - est\_tblpages\_ff)/tblpages::float

ELSE 0

END AS bloat\_ratio, is\_na

-- , (pst).free\_percent + (pst).dead\_tuple\_percent AS real\_frag

FROM (

SELECT ceil( reltuples / ( (bs-page\_hdr)/tpl\_size ) ) + ceil( toasttuples / 4 ) AS est\_tblpages,

ceil( reltuples / ( (bs-page\_hdr)\*fillfactor/(tpl\_size\*100) ) ) + ceil( toasttuples / 4 ) AS est\_tblpages\_ff,

tblpages, fillfactor, bs, tblid, schemaname, tblname, heappages, toastpages, is\_na

-- , stattuple.pgstattuple(tblid) AS pst

FROM (

SELECT

( 4 + tpl\_hdr\_size + tpl\_data\_size + (2\*ma)

- CASE WHEN tpl\_hdr\_size%ma = 0 THEN ma ELSE tpl\_hdr\_size%ma END

- CASE WHEN ceil(tpl\_data\_size)::int%ma = 0 THEN ma ELSE ceil(tpl\_data\_size)::int%ma END

) AS tpl\_size, bs - page\_hdr AS size\_per\_block, (heappages + toastpages) AS tblpages, heappages,

toastpages, reltuples, toasttuples, bs, page\_hdr, tblid, schemaname, tblname, fillfactor, is\_na

FROM (

SELECT

tbl.oid AS tblid, ns.nspname AS schemaname, tbl.relname AS tblname, tbl.reltuples,

tbl.relpages AS heappages, coalesce(toast.relpages, 0) AS toastpages,

coalesce(toast.reltuples, 0) AS toasttuples,

coalesce(substring(

array\_to\_string(tbl.reloptions, ' ')

FROM '%fillfactor=#"\_\_#"%' FOR '#')::smallint, 100) AS fillfactor,

current\_setting('block\_size')::numeric AS bs,

CASE WHEN version()~'mingw32' OR version()~'64-bit|x86\_64|ppc64|ia64|amd64' THEN 8 ELSE 4 END AS ma,

24 AS page\_hdr,

23 + CASE WHEN MAX(coalesce(null\_frac,0)) > 0 THEN ( 7 + count(\*) ) / 8 ELSE 0::int END

+ CASE WHEN tbl.relhasoids THEN 4 ELSE 0 END AS tpl\_hdr\_size,

sum( (1-coalesce(s.null\_frac, 0)) \* coalesce(s.avg\_width, 1024) ) AS tpl\_data\_size,

bool\_or(att.atttypid = 'pg\_catalog.name'::regtype) AS is\_na

FROM pg\_attribute AS att

JOIN pg\_class AS tbl ON att.attrelid = tbl.oid

JOIN pg\_namespace AS ns ON ns.oid = tbl.relnamespace

JOIN pg\_stats AS s ON s.schemaname=ns.nspname

AND s.tablename = tbl.relname AND s.inherited=false AND s.attname=att.attname

LEFT JOIN pg\_class AS toast ON tbl.reltoastrelid = toast.oid

WHERE att.attnum > 0 AND NOT att.attisdropped

AND tbl.relkind = 'r'

GROUP BY 1,2,3,4,5,6,7,8,9,10, tbl.relhasoids

ORDER BY 2,3

) AS s

) AS s2

) AS s3;

select relname, n\_live\_tup, n\_dead\_tup, (n\_dead\_tup/(n\_dead\_tup+n\_live\_tup)::float)\*100 as "% of bloat", last\_autovacuum, last\_autoanalyze from pg\_stat\_all\_tables where (n\_dead\_tup+n\_live\_tup) > 0 and (n\_dead\_tup/(n\_dead\_tup+n\_live\_tup)::float)\*100 > 0;

select name,setting from pg\_settings;

WITH table\_scans as (

SELECT relid,

tables.idx\_scan + tables.seq\_scan as all\_scans,

( tables.n\_tup\_ins + tables.n\_tup\_upd + tables.n\_tup\_del ) as writes,

pg\_relation\_size(relid) as table\_size

FROM pg\_stat\_user\_tables as tables

),

all\_writes as (

SELECT sum(writes) as total\_writes

FROM table\_scans

),

indexes as (

SELECT idx\_stat.relid, idx\_stat.indexrelid,

idx\_stat.schemaname, idx\_stat.relname as tablename,

idx\_stat.indexrelname as indexname,

idx\_stat.idx\_scan,

pg\_relation\_size(idx\_stat.indexrelid) as index\_bytes,

indexdef ~\* 'USING btree' AS idx\_is\_btree

FROM pg\_stat\_user\_indexes as idx\_stat

JOIN pg\_index

USING (indexrelid)

JOIN pg\_indexes as indexes

ON idx\_stat.schemaname = indexes.schemaname

AND idx\_stat.relname = indexes.tablename

AND idx\_stat.indexrelname = indexes.indexname

WHERE pg\_index.indisunique = FALSE

),

index\_ratios AS (

SELECT schemaname, tablename, indexname,

idx\_scan, all\_scans,

round(( CASE WHEN all\_scans = 0 THEN 0.0::NUMERIC

ELSE idx\_scan::NUMERIC/all\_scans \* 100 END),2) as index\_scan\_pct,

writes,

round((CASE WHEN writes = 0 THEN idx\_scan::NUMERIC ELSE idx\_scan::NUMERIC/writes END),2)

as scans\_per\_write,

pg\_size\_pretty(index\_bytes) as index\_size,

pg\_size\_pretty(table\_size) as table\_size,

idx\_is\_btree, index\_bytes

FROM indexes

JOIN table\_scans

USING (relid)

),

index\_groups AS (

SELECT 'Never Used Indexes' as reason, \*, 1 as grp

FROM index\_ratios

WHERE

idx\_scan = 0

and idx\_is\_btree

UNION ALL

SELECT 'Low Scans, High Writes' as reason, \*, 2 as grp

FROM index\_ratios

WHERE

scans\_per\_write <= 1

and index\_scan\_pct < 10

and idx\_scan > 0

and writes > 100

and idx\_is\_btree

UNION ALL

SELECT 'Seldom Used Large Indexes' as reason, \*, 3 as grp

FROM index\_ratios

WHERE

index\_scan\_pct < 5

and scans\_per\_write > 1

and idx\_scan > 0

and idx\_is\_btree

and index\_bytes > 100000000

UNION ALL

SELECT 'High-Write Large Non-Btree' as reason, index\_ratios.\*, 4 as grp

FROM index\_ratios, all\_writes

WHERE

( writes::NUMERIC / ( total\_writes + 1 ) ) > 0.02

AND NOT idx\_is\_btree

AND index\_bytes > 100000000

ORDER BY grp, index\_bytes DESC )

SELECT reason, schemaname, tablename, indexname,

index\_scan\_pct, scans\_per\_write, index\_size, table\_size

FROM index\_groups;

SELECT datname, age(datfrozenxid) FROM pg\_database;

SELECT c.oid::regclass as table\_name,

greatest(age(c.relfrozenxid),age(t.relfrozenxid)) as age

FROM pg\_class c

LEFT JOIN pg\_class t ON c.reltoastrelid = t.oid

WHERE c.relkind IN ('r', 'm');

SELECT

indrelid::regclass AS TableName

,array\_agg(indexrelid::regclass) AS Indexes

FROM pg\_index

GROUP BY

indrelid

,indkey

HAVING COUNT(\*) > 1;

SELECT blocked\_locks.pid AS blocked\_pid,

blocked\_activity.usename AS blocked\_user,

blocking\_locks.pid AS blocking\_pid,

blocking\_activity.usename AS blocking\_user,

blocked\_activity.query AS blocked\_statement,

blocking\_activity.query AS current\_statement\_in\_blocking\_process

FROM pg\_catalog.pg\_locks blocked\_locks

JOIN pg\_catalog.pg\_stat\_activity blocked\_activity ON blocked\_activity.pid = blocked\_locks.pid

JOIN pg\_catalog.pg\_locks blocking\_locks

ON blocking\_locks.locktype = blocked\_locks.locktype

AND blocking\_locks.DATABASE IS NOT DISTINCT FROM blocked\_locks.DATABASE

AND blocking\_locks.relation IS NOT DISTINCT FROM blocked\_locks.relation

AND blocking\_locks.page IS NOT DISTINCT FROM blocked\_locks.page

AND blocking\_locks.tuple IS NOT DISTINCT FROM blocked\_locks.tuple

AND blocking\_locks.virtualxid IS NOT DISTINCT FROM blocked\_locks.virtualxid

AND blocking\_locks.transactionid IS NOT DISTINCT FROM blocked\_locks.transactionid

AND blocking\_locks.classid IS NOT DISTINCT FROM blocked\_locks.classid

AND blocking\_locks.objid IS NOT DISTINCT FROM blocked\_locks.objid

AND blocking\_locks.objsubid IS NOT DISTINCT FROM blocked\_locks.objsubid

AND blocking\_locks.pid != blocked\_locks.pid

JOIN pg\_catalog.pg\_stat\_activity blocking\_activity ON blocking\_activity.pid = blocking\_locks.pid

WHERE NOT blocked\_locks.GRANTED;

Locking session :

SELECT bl.pid AS blocked\_pid,

a.query AS blocking\_statement,

now ( ) - ka.query\_start AS blocking\_duration,

kl.pid AS blocking\_pid,

a.query AS blocked\_statement,

now ( ) - a.query\_start AS blocked\_duration

FROM pg\_catalog.pg\_locks bl

JOIN pg\_catalog.pg\_stat\_activity a ON bl.pid = a.pid

JOIN pg\_catalog.pg\_locks kl

JOIN pg\_catalog.pg\_stat\_activity ka

ON kl.pid = ka.pid

ON bl.transactionid = kl.transactionid

AND bl.pid != kl.pid

WHERE NOT bl.granted;

Blocking query :

SELECT blocked\_locks.pid AS blocked\_pid,

blocked\_activity.usename AS blocked\_user,

blocking\_locks.pid AS blocking\_pid,

blocking\_activity.usename AS blocking\_user,

blocked\_activity.query AS blocked\_statement,

blocking\_activity.query AS current\_statement\_in\_blocking\_process

FROM pg\_catalog.pg\_locks blocked\_locks

JOIN pg\_catalog.pg\_stat\_activity blocked\_activity ON blocked\_activity.pid = blocked\_locks.pid

JOIN pg\_catalog.pg\_locks blocking\_locks

ON blocking\_locks.locktype = blocked\_locks.locktype

AND blocking\_locks.DATABASE IS NOT DISTINCT FROM blocked\_locks.DATABASE

AND blocking\_locks.relation IS NOT DISTINCT FROM blocked\_locks.relation

AND blocking\_locks.page IS NOT DISTINCT FROM blocked\_locks.page

AND blocking\_locks.tuple IS NOT DISTINCT FROM blocked\_locks.tuple

AND blocking\_locks.virtualxid IS NOT DISTINCT FROM blocked\_locks.virtualxid

AND blocking\_locks.transactionid IS NOT DISTINCT FROM blocked\_locks.transactionid

AND blocking\_locks.classid IS NOT DISTINCT FROM blocked\_locks.classid

AND blocking\_locks.objid IS NOT DISTINCT FROM blocked\_locks.objid

AND blocking\_locks.objsubid IS NOT DISTINCT FROM blocked\_locks.objsubid

AND blocking\_locks.pid != blocked\_locks.pid

JOIN pg\_catalog.pg\_stat\_activity blocking\_activity ON blocking\_activity.pid = blocking\_locks.pid WHERE NOT blocked\_locks.GRANTED;

select now()-query\_start as Running\_Since,pid, datname, usename, application\_name, client\_addr , left(query,60) from pg\_stat\_activity where state in ('active','idle in transaction') and (now() - pg\_stat\_activity.query\_start) > interval ‘2 minutes';

DELETE FROM dupes a

WHERE a.ctid <> (SELECT min(b.ctid)

FROM dupes b

WHERE a.key = b.key);

SELECT sum(xact\_commit+xact\_rollback) FROM pg\_stat\_database;

SELECT 'grant select,update,usage on '||c.relname||' to username;' FROM pg\_catalog.pg\_class c

LEFT JOIN pg\_catalog.pg\_namespace n ON n.oid = c.relnamespace

WHERE c.relkind IN ('r','') AND n.nspname='schemaname' AND pg\_catalog.pg\_get\_userbyid(c.relowner)='username';

SELECT n.nspname as "Schema",

c.relname as "Name",

CASE c.relkind WHEN 'r' THEN 'table' WHEN 'v' THEN 'view' WHEN 'S' THEN 'sequence' END as "Type",

pg\_catalog.array\_to\_string(c.relacl, E'\n') AS "Access privileges",

pg\_catalog.array\_to\_string(ARRAY(

SELECT attname || E':\n ' || pg\_catalog.array\_to\_string(attacl, E'\n ')

FROM pg\_catalog.pg\_attribute a

WHERE attrelid = c.oid AND NOT attisdropped AND attacl IS NOT NULL

), E'\n') AS "Column access privileges"

FROM pg\_catalog.pg\_class c

LEFT JOIN pg\_catalog.pg\_namespace n ON n.oid = c.relnamespace

WHERE c.relkind IN ('r') AND pg\_catalog.pg\_get\_userbyid(c.relowner)='username' AND n.nspname='schemaname';

SELECT n.nspname || '.' || p.proname || '(' || pg\_catalog.oidvectortypes(p.proargtypes) || ')' as FunctionName,usename as OWNER FROM pg\_proc p LEFT JOIN pg\_catalog.pg\_namespace n ON n.oid = p.pronamespace, pg\_user u WHERE p.prorettype <> 'pg\_catalog.cstring'::pg\_catalog.regtype AND p.proargtypes[0] <> 'pg\_catalog.cstring'::pg\_catalog.regtype AND pg\_catalog.pg\_function\_is\_visible(p.oid) AND p.proowner=u.usesysid AND n.nspname not in ('pg\_catalog','sys');

select proname||'('||pg\_get\_function\_arguments(pg\_proc.oid)||')' as function\_arguments,usename,nspname from pg\_proc,pg\_user,pg\_namespace where proowner=pg\_user.usesysid and pronamespace=pg\_namespace.oid and usename<>nspname and nspname !~ '^pg\_catalog|^information\_schema|^sys';

SELECT n.nspname as "Schema",

c.relname as "Name",

CASE c.relkind WHEN 'r' THEN 'table' WHEN 'v' THEN 'view' WHEN 'S' THEN 'sequence' WHEN 'f' THEN 'foreign table' END as "Type",

pg\_catalog.array\_to\_string(c.relacl, E'\n') AS "Access privileges",

pg\_catalog.array\_to\_string(ARRAY(

SELECT attname || E':\n ' || pg\_catalog.array\_to\_string(attacl, E'\n ')

FROM pg\_catalog.pg\_attribute a

WHERE attrelid = c.oid AND NOT attisdropped AND attacl IS NOT NULL

), E'\n') AS "Column access privileges"

FROM pg\_catalog.pg\_class c

LEFT JOIN pg\_catalog.pg\_namespace n ON n.oid = c.relnamespace

WHERE c.relkind IN ('r', 'v', 'S', 'f')

AND n.nspname !~ '^pg\_' AND pg\_catalog.pg\_table\_is\_visible(c.oid) and pg\_catalog.pg\_get\_userbyid(c.relowner)='owner'

ORDER BY 1, 2;

select 'grant execute on procedure "CBF"."'||proname||'"('||pg\_get\_function\_arguments(oid)||') to cbf\_ctrl\_user;' from pg\_proc where pronamespace=' <oid of schema>' ;

SELECT

pgClass.relname AS tableName,

pgClass.reltuples AS rowCount

FROM

pg\_class pgClass

LEFT JOIN

pg\_namespace pgNamespace ON (pgNamespace.oid = pgClass.relnamespace)

WHERE

pgNamespace.nspname NOT IN ('pg\_catalog', 'information\_schema') AND

pgClass.relkind='r';

SELECT n.nspname as "Schema",

count(c.relname) as "Name"

FROM pg\_catalog.pg\_class c

LEFT JOIN pg\_catalog.pg\_namespace n ON n.oid = c.relnamespace

WHERE c.relkind IN ('r','')

AND n.nspname <> 'pg\_catalog'

AND n.nspname <> 'information\_schema'

AND n.nspname !~ '^pg\_toast'

AND pg\_catalog.pg\_table\_is\_visible(c.oid)

group by n.nspname;

SELECT c.relname, pg\_catalog.array\_to\_string(c.reloptions || array(select 'toast.' || x from pg\_catalog.unnest(tc.reloptions) x), ', ')

FROM pg\_catalog.pg\_class c

LEFT JOIN pg\_catalog.pg\_class tc ON (c.reltoastrelid = tc.oid)

WHERE c.relname = 'test'

SELECT 'alter table "'||c.relname||'" rename to '||lower(c.relname)||';'

FROM pg\_catalog.pg\_class c

LEFT JOIN pg\_catalog.pg\_namespace n ON n.oid = c.relnamespace

WHERE c.relkind ='r'

AND n.nspname='schemaname'

ORDER BY 1;

SELECT

'alter table "'||c.relname||'" rename "'||a.attname||'" to '||lower(a.attname)||';'

FROM

pg\_class c

JOIN pg\_attribute a ON a.attrelid = c.oid

JOIN pg\_type t ON a.atttypid = t.oid

LEFT JOIN pg\_catalog.pg\_constraint r ON c.oid = r.conrelid

AND r.conname = a.attname

WHERE

c.relnamespace = (select oid from pg\_namespace where nspname='schemaname')

AND a.attnum > 0 AND c.relkind in ('r', 'p')

AND c.relname = 'table\_name'

ORDER BY a.attnum

SELECT

'alter table "'||c.relname||'" rename "'||a.attname||'" to '||lower(a.attname)||';'

FROM

pg\_class c

JOIN pg\_attribute a ON a.attrelid = c.oid

JOIN pg\_type t ON a.atttypid = t.oid

LEFT JOIN pg\_catalog.pg\_constraint r ON c.oid = r.conrelid

AND r.conname = a.attname

WHERE

c.relnamespace = (select oid from pg\_namespace where nspname='schemaname')

AND a.attnum > 0

AND c.relkind in ('r', 'p')

ORDER BY a.attnum

SELECT c2.relname, i.indisprimary, i.indisunique, i.indisvalid, pg\_catalog.pg\_get\_indexdef(i.indexrelid, 0, true),

pg\_catalog.pg\_get\_constraintdef(con.oid, true), contype

FROM pg\_catalog.pg\_class c, pg\_catalog.pg\_class c2, pg\_catalog.pg\_index i

LEFT JOIN pg\_catalog.pg\_constraint con ON (conrelid = i.indrelid AND conindid = i.indexrelid AND contype IN ('p'))

WHERE c.relnamespace=(select oid from pg\_namespace where nspname='public') AND c.oid = i.indrelid AND i.indexrelid = c2.oid

ORDER BY i.indisprimary DESC, i.indisunique DESC, c2.relname;

SELECT n.nspname as "Schema",

c.relname as "Name",

CASE c.relkind WHEN 'r' THEN 'table' WHEN 'v' THEN 'view' WHEN 'm' THEN 'materialized view' WHEN 'i' THEN 'index' WHEN 'S' THEN 'sequence' WHEN 's' THEN 'special' WHEN 'f' THEN 'foreign table' END as "Type",

pg\_catalog.pg\_get\_userbyid(c.relowner) as "Owner"

FROM pg\_catalog.pg\_class c

LEFT JOIN pg\_catalog.pg\_namespace n ON n.oid = c.relnamespace

WHERE c.relkind IN ('S','')

AND n.nspname='schemaname'

ORDER BY 1,2;

SELECT r.conname

FROM pg\_catalog.pg\_constraint r

WHERE r.connamespace = (select oid from pg\_namespace where nspname='public') AND r.contype = 'c'

ORDER BY 1;

SELECT conname,

pg\_catalog.pg\_get\_constraintdef(r.oid, true) as condef

FROM pg\_catalog.pg\_constraint r

WHERE r.connamespace=(select oid from pg\_namespace where nspname='public') AND r.contype = 'f' ORDER BY 1;

SELECT conname, conrelid::regclass, conindid::regclass,

pg\_catalog.pg\_get\_constraintdef(r.oid, true) as condef

FROM pg\_catalog.pg\_constraint r

WHERE r.connamespace=(select oid from pg\_namespace where nspname='public') AND r.contype = 'f' ORDER BY 1;

select s.relname as "Sequence", n.nspname as "schema", t.relname as "Owned by table", a.attname as "Owned by column"

from pg\_class s

join pg\_depend d on d.objid=s.oid and d.classid='pg\_class'::regclass and d.refclassid='pg\_class'::regclass

join pg\_class t on t.oid=d.refobjid

join pg\_namespace n on n.oid=t.relnamespace

join pg\_attribute a on a.attrelid=t.oid and a.attnum=d.refobjsubid

where s.relkind='S'