PostgreSQL DBA Daily Checklist

The daily checklist of a PostgreSQL DBA

We often get this question, What are the most important things a PostgreSQL DBA should do to guarantee optimal performance and reliability, Do we have checklist for PostgreSQL DBAs to follow daily ? Since we are getting this question too often, Thought let's note it as blog post and share with community of PostgreSQL ecosystem. The only objective this post is to share the information, Please don't consider this as a run-book or recommendation from MinervaDB PostgreSQL support. We at MinervaDB are not accountable of any negative performance in you PostgreSQL performance with running these scripts in production database infrastructure of your business, The following is a simple daily checklist for PostgreSQL DBA:

Task 1: Check that all the PostgreSQL instances are up and operational:

Shell

```
pgrep -u postgres -fa -- -D
```

What if you have several instances of PostgreSQL are running:

Shell

```
pgrep -fa -- -D |grep postgres
```

Task 2: Monitoring PostgreSQL logsRecord PostgreSQL error logs: Open postgresql.conf configuration file, Under the **ERROR REPORTING AND LOGGING** section of the file, use following config parameters:

Shell

```
log_destination = 'stderr'
logging_collector = on
log_directory = 'pg_log'
log_filename = 'postgresql-%Y-%m-%d_%H%M%S.log'
log_truncate_on_rotation = off
log_rotation_age = 1d
```

```
log_min_duration_statement = 0
log_connections = on
log_duration = on
log_hostname = on
log_timezone = 'UTC'
```

Save the **postgresql.conf** file and restart the PostgreSQL server:

Shell

```
sudo service postgresql restart
```

Task 3: Confirm PostgreSQL backup completed successfully

Use backup logs (possible only with PostgreSQL logical backup) to audit backup quality:

Shell

```
$ pg_dumpall > /backup-path/pg-backup-dump.sql > /var/log/postgres/pg-backup-dump.log
```

Task 4: Monitoring PostgreSQL Database Size:

Shell

```
select datname, pg_size_pretty(pg_database_size(datname)) from pg_database order by pg_database_size(datname);
```

Task 5: Monitor all PostgreSQL queries running (please repeat this task every 90 minutes during business / peak hours):

Shell

```
SELECT pid, age(clock_timestamp(), query_start), usename, query
FROM pg_stat_activity
WHERE query != '<IDLE>' AND query NOT ILIKE '%pg_stat_activity%'
ORDER BY query_start desc;
```

Task 6: Inventory of indexes in PostgreSQL database:

Shell

```
select
    t.relname as table_name,
```

```
    i.relname as index_name,
    string_agg(a.attname, ',') as column_name
from
    pg_class t,
    pg_class i,
    pg_index ix,
    pg_attribute a
where
    t.oid = ix.indrelid
    and i.oid = ix.indexrelid
    and a.attrelid = t.oid
    and a.attnum = ANY(ix.indkey)
    and t.relkind = 'r'
    and t.relname not like 'pg_%'
group by
    t.relname,
    i.relname
order by
    t.relname,
    i.relname;
```

Task 7: Finding the largest databases in your PostgreSQL cluster:

Shell

```
SELECT d.datname as Name,  pg_catalog.pg_get_userbyid(d.datdba) as Owner,
    CASE WHEN pg_catalog.has_database_privilege(d.datname, 'CONNECT')
        THEN pg_catalog.pg_size_pretty(pg_catalog.pg_database_size(d.datname))
        ELSE 'No Access'
    END as Size
FROM pg_catalog.pg_database d
    order by
    CASE WHEN pg_catalog.has_database_privilege(d.datname, 'CONNECT')
        THEN pg_catalog.pg_database_size(d.datname)
        ELSE NULL
    END desc -- nulls first
```

```
12    LIMIT 20
```

Task 8: when you are suspecting some serious performance bottleneck in PostgreSQL ? Especially when you suspecting transactions blocking each other:

Shell

```
1  WITH RECURSIVE l AS (
2    SELECT pid, locktype, mode, granted,
3  ROW(locktype,database,relation,page,tuple,virtualxid,transactionid,classid,objid,objsubi
4  d) obj
5    FROM pg_locks
6  ), pairs AS (
7    SELECT w.pid waiter, l.pid locker, l.obj, l.mode
8    FROM l w
9    JOIN l ON l.obj IS NOT DISTINCT FROM w.obj AND l.locktype=w.locktype AND NOT
10 l.pid=w.pid AND l.granted
11   WHERE NOT w.granted
   ), tree AS (
12   SELECT l.locker pid, l.locker root, NULL::record obj, NULL AS mode, 0 lvl, locker::text
   path, array_agg(l.locker) OVER () all_pids
13   FROM ( SELECT DISTINCT locker FROM pairs l WHERE NOT EXISTS (SELECT 1 F
14 ROM pairs WHERE waiter=l.locker) ) l
     UNION ALL
15   SELECT w.waiter pid, tree.root, w.obj, w.mode, tree.lvl+1, tree.path||'.'||w.waiter, all_pi
16 ds || array_agg(w.waiter) OVER ()
17   FROM tree JOIN pairs w ON tree.pid=w.locker AND NOT w.waiter = ANY ( all_pids )
18 )
19 SELECT (clock_timestamp() - a.xact_start)::interval(3) AS ts_age,
20       replace(a.state, 'idle in transaction', 'idletx') state,
21       (clock_timestamp() - state_change)::interval(3) AS change_age,
22       a.datname,tree.pid,a.usename,a.client_addr,lvl,
23       (SELECT count(*) FROM tree p WHERE p.path ~ ('^'||tree.path) AND NOT p.path=t
24 ree.path) blocked,
25       repeat(' .', lvl)||' '||left(regexp_replace(query, 's+', ' ', 'g'),100) query
   FROM tree
```

```
JOIN pg_stat_activity a USING (pid)
ORDER BY path;
```

Task 9: Identify bloated Tables in PostgreSQL :

Shell

```
WITH constants AS (
    -- define some constants for sizes of things
    -- for reference down the query and easy maintenance
    SELECT current_setting('block_size')::numeric AS bs, 23 AS hdr, 8 AS ma
),
no_stats AS (
    -- screen out table who have attributes
    -- which dont have stats, such as JSON
    SELECT table_schema, table_name,
        n_live_tup::numeric as est_rows,
        pg_table_size(relid)::numeric as table_size
    FROM information_schema.columns
        JOIN pg_stat_user_tables as psut
          ON table_schema = psut.schemaname
          AND table_name = psut.relname
        LEFT OUTER JOIN pg_stats
        ON table_schema = pg_stats.schemaname
            AND table_name = pg_stats.tablename
            AND column_name = attname
    WHERE attname IS NULL
        AND table_schema NOT IN ('pg_catalog', 'information_schema')
    GROUP BY table_schema, table_name, relid, n_live_tup
),
null_headers AS (
    -- calculate null header sizes
    -- omitting tables which dont have complete stats
    -- and attributes which aren't visible
    SELECT
        hdr+1+(sum(case when null_frac <> 0 THEN 1 else 0 END)/8) as nullhdr,
```

```
30        SUM((1-null_frac)*avg_width) as datawidth,
31        MAX(null_frac) as maxfracsum,
32        schemaname,
33        tablename,
34        hdr, ma, bs
35      FROM pg_stats CROSS JOIN constants
36        LEFT OUTER JOIN no_stats
37          ON schemaname = no_stats.table_schema
38          AND tablename = no_stats.table_name
39      WHERE schemaname NOT IN ('pg_catalog', 'information_schema')
40        AND no_stats.table_name IS NULL
41        AND EXISTS ( SELECT 1
42          FROM information_schema.columns
43            WHERE schemaname = columns.table_schema
44              AND tablename = columns.table_name )
45      GROUP BY schemaname, tablename, hdr, ma, bs
46  ),
47  data_headers AS (
48      -- estimate header and row size
49      SELECT
50        ma, bs, hdr, schemaname, tablename,
51        (datawidth+(hdr+ma-(case when hdr%ma=0 THEN ma ELSE hdr%ma END)))::n
52  umeric AS datahdr,
53        (maxfracsum*(nullhdr+ma-(case when nullhdr%ma=0 THEN ma ELSE nullhdr%
54  ma END))) AS nullhdr2
55      FROM null_headers
56  ),
57  table_estimates AS (
58      -- make estimates of how large the table should be
59      -- based on row and page size
60      SELECT schemaname, tablename, bs,
61        reltuples::numeric as est_rows, relpages * bs as table_bytes,
62      CEIL((reltuples*
63            (datahdr + nullhdr2 + 4 + ma -
64              (CASE WHEN datahdr%ma=0
```

```
                THEN ma ELSE datahdr%ma END)
              )/(bs-20))) * bs AS expected_bytes,
        reltoastrelid
    FROM data_headers
        JOIN pg_class ON tablename = relname
        JOIN pg_namespace ON relnamespace = pg_namespace.oid
            AND schemaname = nspname
    WHERE pg_class.relkind = 'r'
),
estimates_with_toast AS (
    -- add in estimated TOAST table sizes
    -- estimate based on 4 toast tuples per page because we dont have
    -- anything better.  also append the no_data tables
    SELECT schemaname, tablename,
        TRUE as can_estimate,
        est_rows,
        table_bytes + ( coalesce(toast.relpages, 0) * bs ) as table_bytes,
        expected_bytes + ( ceil( coalesce(toast.reltuples, 0) / 4 ) * bs ) as expected_byte
s
    FROM table_estimates LEFT OUTER JOIN pg_class as toast
        ON table_estimates.reltoastrelid = toast.oid
            AND toast.relkind = 't'
),
table_estimates_plus AS (
-- add some extra metadata to the table data
-- and calculations to be reused
-- including whether we cant estimate it
-- or whether we think it might be compressed
    SELECT current_database() as databasename,
            schemaname, tablename, can_estimate,
            est_rows,
            CASE WHEN table_bytes > 0
                THEN table_bytes::NUMERIC
                ELSE NULL::NUMERIC END
                AS table_bytes,
```

```
100           CASE WHEN expected_bytes > 0
101              THEN expected_bytes::NUMERIC
102              ELSE NULL::NUMERIC END
103                 AS expected_bytes,
104           CASE WHEN expected_bytes > 0 AND table_bytes > 0
105              AND expected_bytes <= table_bytes
106              THEN (table_bytes - expected_bytes)::NUMERIC
107              ELSE 0::NUMERIC END AS bloat_bytes
108     FROM estimates_with_toast
109     UNION ALL
110     SELECT current_database() as databasename,
111         table_schema, table_name, FALSE,
112         est_rows, table_size,
113         NULL::NUMERIC, NULL::NUMERIC
114     FROM no_stats
115 ),
116 bloat_data AS (
117     -- do final math calculations and formatting
118     select current_database() as databasename,
119         schemaname, tablename, can_estimate,
120         table_bytes, round(table_bytes/(1024^2)::NUMERIC,3) as table_mb,
121         expected_bytes, round(expected_bytes/(1024^2)::NUMERIC,3) as expected_mb,
122         round(bloat_bytes*100/table_bytes) as  pct_bloat,
123         round(bloat_bytes/(1024::NUMERIC^2),2) as mb_bloat,
124         table_bytes, expected_bytes, est_rows
125     FROM table_estimates_plus
126 )
127 -- filter output for bloated tables
128 SELECT databasename, schemaname, tablename,
129     can_estimate,
130     est_rows,
131     pct_bloat, mb_bloat,
132     table_mb
133 FROM bloat_data
134 -- this where clause defines which tables actually appear
```

```
-- in the bloat chart
-- example below filters for tables which are either 50%
-- bloated and more than 20mb in size, or more than 25%
-- bloated and more than 4GB in size
WHERE ( pct_bloat >= 50 AND mb_bloat >= 10 )
   OR ( pct_bloat >= 25 AND mb_bloat >= 1000 )
ORDER BY pct_bloat DESC;
```

Task 10: Identify bloated indexes in PostgreSQL :

Shell

```
-- btree index stats query
-- estimates bloat for btree indexes
WITH btree_index_atts AS (
    SELECT nspname,
        indexclass.relname as index_name,
        indexclass.reltuples,
        indexclass.relpages,
        indrelid, indexrelid,
        indexclass.relam,
        tableclass.relname as tablename,
        regexp_split_to_table(indkey::text, ' ')::smallint AS attnum,
        indexrelid as index_oid
    FROM pg_index
    JOIN pg_class AS indexclass ON pg_index.indexrelid = indexclass.oid
    JOIN pg_class AS tableclass ON pg_index.indrelid = tableclass.oid
    JOIN pg_namespace ON pg_namespace.oid = indexclass.relnamespace
    JOIN pg_am ON indexclass.relam = pg_am.oid
    WHERE pg_am.amname = 'btree' and indexclass.relpages > 0
        AND nspname NOT IN ('pg_catalog','information_schema')
    ),
index_item_sizes AS (
    SELECT
    ind_atts.nspname, ind_atts.index_name,
    ind_atts.reltuples, ind_atts.relpages, ind_atts.relam,
```

```
    indrelid AS table_oid, index_oid,
    current_setting('block_size')::numeric AS bs,
    8 AS maxalign,
    24 AS pagehdr,
    CASE WHEN max(coalesce(pg_stats.null_frac,0)) = 0
       THEN 2
       ELSE 6
    END AS index_tuple_hdr,
    sum( (1-coalesce(pg_stats.null_frac, 0)) * coalesce(pg_stats.avg_width, 1024) ) AS
nulldatawidth
    FROM pg_attribute
    JOIN btree_index_atts AS ind_atts ON pg_attribute.attrelid = ind_atts.indexrelid AN
D pg_attribute.attnum = ind_atts.attnum
    JOIN pg_stats ON pg_stats.schemaname = ind_atts.nspname
        -- stats for regular index columns
        AND ( (pg_stats.tablename = ind_atts.tablename AND pg_stats.attname = pg_c
atalog.pg_get_indexdef(pg_attribute.attrelid, pg_attribute.attnum, TRUE))
        -- stats for functional indexes
        OR   (pg_stats.tablename = ind_atts.index_name AND pg_stats.attname = pg_a
ttribute.attname))
    WHERE pg_attribute.attnum > 0
    GROUP BY 1, 2, 3, 4, 5, 6, 7, 8, 9
),
index_aligned_est AS (
    SELECT maxalign, bs, nspname, index_name, reltuples,
       relpages, relam, table_oid, index_oid,
       coalesce (
          ceil (
             reltuples * ( 6
                + maxalign
                - CASE
                   WHEN index_tuple_hdr%maxalign = 0 THEN maxalign
                   ELSE index_tuple_hdr%maxalign
                  END
                + nulldatawidth
```

```
58                  + maxalign
59                  - CASE /* Add padding to the data to align on MAXALIGN */
60                    WHEN nulldatawidth::integer%maxalign = 0 THEN maxalign
61                    ELSE nulldatawidth::integer%maxalign
62                  END
63              )::numeric
64             / ( bs - pagehdr::NUMERIC )
65             +1 )
66         , 0 )
67       as expected
68     FROM index_item_sizes
69 ),
70 raw_bloat AS (
71     SELECT current_database() as dbname, nspname, pg_class.relname AS table_na
72 me, index_name,
73         bs*(index_aligned_est.relpages)::bigint AS totalbytes, expected,
74         CASE
75           WHEN index_aligned_est.relpages <= expected
76             THEN 0
77             ELSE bs*(index_aligned_est.relpages-expected)::bigint
78           END AS wastedbytes,
79         CASE
80           WHEN index_aligned_est.relpages <= expected
81             THEN 0
82             ELSE bs*(index_aligned_est.relpages-expected)::bigint * 100 / (bs*(index_al
83 igned_est.relpages)::bigint)
84           END AS realbloat,
85         pg_relation_size(index_aligned_est.table_oid) as table_bytes,
86         stat.idx_scan as index_scans
87     FROM index_aligned_est
88     JOIN pg_class ON pg_class.oid=index_aligned_est.table_oid
89     JOIN pg_stat_user_indexes AS stat ON index_aligned_est.index_oid = stat.indexrel
90 id
91 ),
92 format_bloat AS (
```

```
SELECT dbname as database_name, nspname as schema_name, table_name, index
_name,
        round(realbloat) as bloat_pct, round(wastedbytes/(1024^2)::NUMERIC) as bloat_
mb,
        round(totalbytes/(1024^2)::NUMERIC,3) as index_mb,
        round(table_bytes/(1024^2)::NUMERIC,3) as table_mb,
        index_scans
FROM raw_bloat
)
    -- final query outputting the bloated indexes
    -- change the where and order by to change
    -- what shows up as bloated
    SELECT *
    FROM format_bloat
    WHERE ( bloat_pct > 50 and bloat_mb > 10 )
    ORDER BY bloat_mb DESC;
```

Task 11: Monitor blocked and blocking activities in PostgreSQL:

Shell

```
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
```

```
16        AND blocking_locks.virtualxid IS NOT DISTINCT FROM blocked_locks.virtualxid
17        AND blocking_locks.transactionid IS NOT DISTINCT FROM blocked_locks.transac
18  tionid
19        AND blocking_locks.classid IS NOT DISTINCT FROM blocked_locks.classid
20        AND blocking_locks.objid IS NOT DISTINCT FROM blocked_locks.objid
21        AND blocking_locks.objsubid IS NOT DISTINCT FROM blocked_locks.objsubid
22        AND blocking_locks.pid != blocked_locks.pid
23
      JOIN pg_catalog.pg_stat_activity blocking_activity ON blocking_activity.pid = blockin
    g_locks.pid
     WHERE NOT blocked_locks.granted;
```

Task 12: Monitoring PostgreSQL Disk I/O performance:

Shell

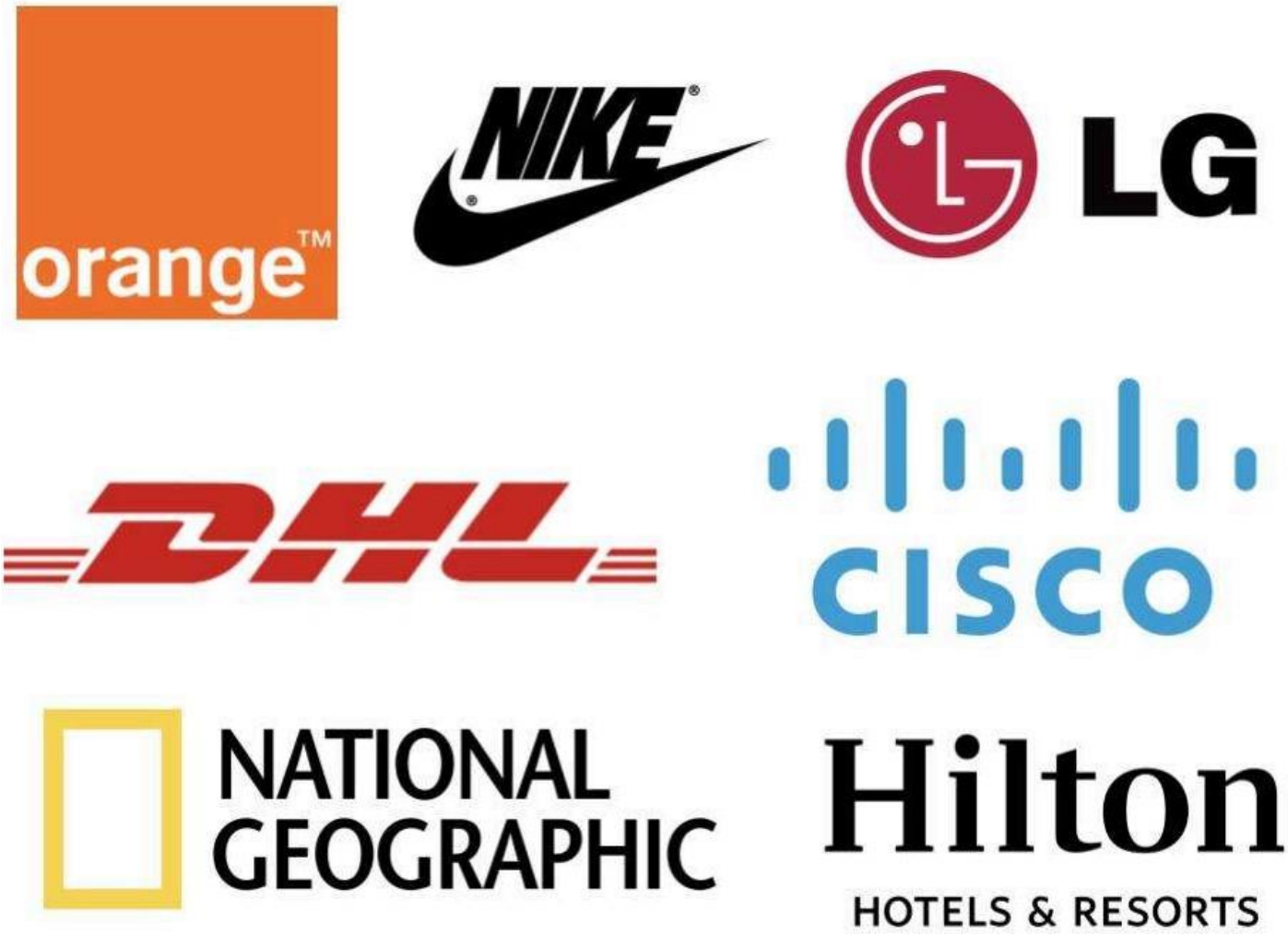
```
1  -- perform a "select pg_stat_reset();" when you want to reset counter statistics
2  with
3  all_tables as
4  (
5  SELECT  *
6  FROM    (
7     SELECT  'all'::text as table_name,
8         sum( (coalesce(heap_blks_read,0) + coalesce(idx_blks_read,0) + coalesce(toast_
   blks_read,0) + coalesce(tidx_blks_read,0)) ) as from_disk,
9         sum( (coalesce(heap_blks_hit,0)  + coalesce(idx_blks_hit,0)  + coalesce(toast_blk
   s_hit,0)  + coalesce(tidx_blks_hit,0))  ) as from_cache
10    FROM    pg_statio_all_tables  --> change to pg_statio_USER_tables if you want to c
   heck only user tables (excluding postgres's own tables)
11    ) a
12 WHERE   (from_disk + from_cache) > 0 -- discard tables without hits
13 ),
14 tables as
15 (
16 SELECT  *
17 FROM    (
```

```
18    SELECT  relname as table_name,
19        ( (coalesce(heap_blks_read,0) + coalesce(idx_blks_read,0) + coalesce(toast_blks
    _read,0) + coalesce(tidx_blks_read,0)) ) as from_disk,
20        ( (coalesce(heap_blks_hit,0)  + coalesce(idx_blks_hit,0)  + coalesce(toast_blks_hi
    t,0)  + coalesce(tidx_blks_hit,0))  ) as from_cache
21    FROM    pg_statio_all_tables --> change to pg_statio_USER_tables if you want to ch
    eck only user tables (excluding postgres's own tables)
22    ) a
23 WHERE   (from_disk + from_cache) > 0 -- discard tables without hits
24 )
25 SELECT  table_name as "table name",
26    from_disk as "disk hits",
27    round((from_disk::numeric / (from_disk + from_cache)::numeric)*100.0,2) as "% disk
28 hits",
29    round((from_cache::numeric / (from_disk + from_cache)::numeric)*100.0,2) as "% ca
30 che hits",
31    (from_disk + from_cache) as "total hits"
   FROM    (SELECT * FROM all_tables UNION ALL SELECT * FROM tables) a
   ORDER   BY (case when table_name = 'all' then 0 else 1 end), from_disk desc;
```

References

- https://www.postgresql.org/developer/related-projects/
- https://www.postgresql.org/community/
- https://github.com/pgexperts

☛ MinervaDB is trusted by top companies worldwide


orange™
NIKE
LG
DHL
CISCO
NATIONAL GEOGRAPHIC
Hilton
HOTELS & RESORTS