

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
1 <code>pgrep -u postgres -fa -- -D</code>	

What if you have several instances of PostgreSQL are running:

	Shell
1 <code>pgrep -fa -- -D grep postgres</code>	

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
1 <code>log_destination = 'stderr'</code>	
2 <code>logging_collector = on</code>	
3 <code>log_directory = 'pg_log'</code>	
4 <code>log_filename = 'postgresql-%Y-%m-%d_%H%M%S.log'</code>	
5 <code>log_truncate_on_rotation = off</code>	
6 <code>log_rotation_age = 1d</code>	

```
7 log_min_duration_statement = 0
8 log_connections = on
9 log_duration = on
10 log_hostname = on
11 log_timezone = 'UTC'
```

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

Shell

```
1 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

```
1 $ pg_dumpall > /backup-path/pg-backup-dump.sql > /var/log/postgres/pg-backup-dump.log
```

Task 4: Monitoring PostgreSQL Database Size:

Shell

```
1 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

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

Task 6: Inventory of indexes in PostgreSQL database:

Shell

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

```
3    i.relname as index_name,
4    string_agg(a.attname, ',') as column_name
5  from
6    pg_class t,
7    pg_class i,
8    pg_index ix,
9    pg_attribute a
10 where
11    t.oid = ix.indrelid
12    and i.oid = ix.indexrelid
13    and a.attrelid = t.oid
14    and a.attnum = ANY(ix.indkey)
15    and t.relkind = 'r'
16    and t.relname not like 'pg_%'
17 group by
18    t.relname,
19    i.relname
20 order by
21    t.relname,
22    i.relname;
```

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

Shell

```
1  SELECT d.datname as Name, pg_catalog.pg_get_userbyid(d.datdba) as Owner,
2     CASE WHEN pg_catalog.has_database_privilege(d.datname, 'CONNECT')
3         THEN pg_catalog.pg_size_pretty(pg_catalog.pg_database_size(d.datname))
4         ELSE 'No Access'
5     END as Size
6  FROM pg_catalog.pg_database d
7  order by
8     CASE WHEN pg_catalog.has_database_privilege(d.datname, 'CONNECT')
9         THEN pg_catalog.pg_database_size(d.datname)
10        ELSE NULL
11  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 I 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 I w
9    JOIN I 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
12   ), tree AS (
13     SELECT l.locker pid, l.locker root, NULL::record obj, NULL AS mode, 0 lvl, locker::text
14     path, array_agg(l.locker) OVER () all_pids
15   FROM ( SELECT DISTINCT locker FROM pairs I WHERE NOT EXISTS (SELECT 1 F
16   ROM pairs WHERE waiter=l.locker) ) I
17   UNION ALL
18   SELECT w.waiter pid, tree.root, w.obj, w.mode, tree.lvl+1, tree.path||'.'||w.waiter, all_pi
19   ds || array_agg(w.waiter) OVER ()
20   FROM tree JOIN pairs w ON tree.pid=w.locker AND NOT w.waiter = ANY ( all_pids )
21   )
22   SELECT (clock_timestamp() - a.xact_start)::interval(3) AS ts_age,
23   replace(a.state, 'idle in transaction', 'idletx') state,
24   (clock_timestamp() - state_change)::interval(3) AS change_age,
25   a.datname,tree.pid,a.username,a.client_addr,lvl,
26   (SELECT count(*) FROM tree p WHERE p.path ~ ('^'||tree.path) AND NOT p.path=t
27   ree.path) blocked,
28   repeat(' ', lvl)||' '||left(regexp_replace(query, 's+', ' ', 'g'),100) query
29   FROM tree

```

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

Task 9: Identify bloated Tables in PostgreSQL :

Shell

```
1  WITH constants AS (
2      -- define some constants for sizes of things
3      -- for reference down the query and easy maintenance
4      SELECT current_setting('block_size')::numeric AS bs, 23 AS hdr, 8 AS ma
5  ),
6  no_stats AS (
7      -- screen out table who have attributes
8      -- which dont have stats, such as JSON
9      SELECT table_schema, table_name,
10         n_live_tup::numeric as est_rows,
11         pg_table_size(relid)::numeric as table_size
12  FROM information_schema.columns
13      JOIN pg_stat_user_tables as psut
14         ON table_schema = psut.schemaname
15         AND table_name = psut.relname
16      LEFT OUTER JOIN pg_stats
17         ON table_schema = pg_stats.schemaname
18         AND table_name = pg_stats.tablename
19         AND column_name = attname
20  WHERE attname IS NULL
21         AND table_schema NOT IN ('pg_catalog', 'information_schema')
22  GROUP BY table_schema, table_name, relid, n_live_tup
23 ),
24 null_headers AS (
25     -- calculate null header sizes
26     -- omitting tables which dont have complete stats
27     -- and attributes which aren't visible
28     SELECT
29         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

```

```

65         THEN ma ELSE datahdr%ma END)
66     )/(bs-20))) * bs AS expected_bytes,
67     reltoastrelid
68 FROM data_headers
69     JOIN pg_class ON tablename = relname
70     JOIN pg_namespace ON relnamespace = pg_namespace.oid
71     AND schemaname = nspname
72 WHERE pg_class.relkind = 'r'
73 ),
74 estimates_with_toast AS (
75     -- add in estimated TOAST table sizes
76     -- estimate based on 4 toast tuples per page because we dont have
77     -- anything better. also append the no_data tables
78     SELECT schemaname, tablename,
79         TRUE as can_estimate,
80         est_rows,
81         table_bytes + ( coalesce(toast.relpages, 0) * bs ) as table_bytes,
82         expected_bytes + ( ceil( coalesce(toast.reltuples, 0) / 4 ) * bs ) as expected_byte
83 s
84 FROM table_estimates LEFT OUTER JOIN pg_class as toast
85     ON table_estimates.reltoastrelid = toast.oid
86     AND toast.relkind = 't'
87 ),
88 table_estimates_plus AS (
89     -- add some extra metadata to the table data
90     -- and calculations to be reused
91     -- including whether we cant estimate it
92     -- or whether we think it might be compressed
93     SELECT current_database() as databasename,
94         schemaname, tablename, can_estimate,
95         est_rows,
96         CASE WHEN table_bytes > 0
97             THEN table_bytes::NUMERIC
98             ELSE NULL::NUMERIC END
99         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
```



```

135 -- in the bloat chart
136 -- example below filters for tables which are either 50%
137 -- bloated and more than 20mb in size, or more than 25%
138 -- 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

```

1  -- btree index stats query
2  -- estimates bloat for btree indexes
3  WITH btree_index_atts AS (
4      SELECT nspname,
5             indexclass.relname as index_name,
6             indexclass.reltuples,
7             indexclass.relpages,
8             indrelid, indexrelid,
9             indexclass.relam,
10            tableclass.relname as tablename,
11            regexp_split_to_table(indkey::text, ' ')::smallint AS attnum,
12            indexrelid as index_oid
13  FROM pg_index
14  JOIN pg_class AS indexclass ON pg_index.indexrelid = indexclass.oid
15  JOIN pg_class AS tableclass ON pg_index.indrelid = tableclass.oid
16  JOIN pg_namespace ON pg_namespace.oid = indexclass.relnamespace
17  JOIN pg_am ON indexclass.relam = pg_am.oid
18  WHERE pg_am.amname = 'btree' and indexclass.relpages > 0
19         AND nspname NOT IN ('pg_catalog','information_schema')
20  ),
21  index_item_sizes AS (
22      SELECT
23          ind_atts.nspname, ind_atts.index_name,
24          ind_atts.reltuples, ind_atts.relpages, ind_atts.relam,

```

```

25     indrelid AS table_oid, index_oid,
26     current_setting('block_size')::numeric AS bs,
27     8 AS maxalign,
28     24 AS pagehdr,
29     CASE WHEN max(coalesce(pg_stats.null_frac,0)) = 0
30         THEN 2
31         ELSE 6
32     END AS index_tuple_hdr,
33     sum( (1-coalesce(pg_stats.null_frac, 0)) * coalesce(pg_stats.avg_width, 1024) ) AS
34 nulldatawidth
35     FROM pg_attribute
36     JOIN btree_index_atts AS ind_atts ON pg_attribute.attrelid = ind_atts.indexrelid AND
37     pg_attribute.attnum = ind_atts.attnum
38     JOIN pg_stats ON pg_stats.schemaname = ind_atts.nspname
39     -- stats for regular index columns
40     AND ( (pg_stats.tablename = ind_atts.tablename AND pg_stats.attnum = pg_c
41 atalog.pg_get_indexdef(pg_attribute.attrelid, pg_attribute.attnum, TRUE))
42     -- stats for functional indexes
43     OR (pg_stats.tablename = ind_atts.index_name AND pg_stats.attnum = pg_a
44 ttribute.attnum))
45     WHERE pg_attribute.attnum > 0
46     GROUP BY 1, 2, 3, 4, 5, 6, 7, 8, 9
47 ),
48 index_aligned_est AS (
49     SELECT maxalign, bs, nspname, index_name, reltuples,
50     relpages, relam, table_oid, index_oid,
51     coalesce (
52         ceil (
53             reltuples * ( 6
54             + maxalign
55             - CASE
56                 WHEN index_tuple_hdr%maxalign = 0 THEN maxalign
57                 ELSE index_tuple_hdr%maxalign
58             END
59             + 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_name,
72     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_aligned_est.relpages)::bigint)
83     END AS realbloat,
84     pg_relation_size(index_aligned_est.table_oid) as table_bytes,
85     stat.idx_scan as index_scans
86 FROM index_aligned_est
87 JOIN pg_class ON pg_class.oid=index_aligned_est.table_oid
88 JOIN pg_stat_user_indexes AS stat ON index_aligned_est.index_oid = stat.indexrelid
89 id
90 ),
91 ),
92 format_bloat AS (

```

```

93 SELECT dbname as database_name, nspname as schema_name, table_name, index
94 _name,
95     round(realbloat) as bloat_pct, round(wastedbytes/(1024^2)::NUMERIC) as bloat_
96 mb,
97     round(totalbytes/(1024^2)::NUMERIC,3) as index_mb,
98     round(table_bytes/(1024^2)::NUMERIC,3) as table_mb,
99     index_scans
100 FROM raw_bloat
101 )
    -- 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

```

1 SELECT blocked_locks.pid AS blocked_pid,
2     blocked_activity.username AS blocked_user,
3     blocking_locks.pid AS blocking_pid,
4     blocking_activity.username AS blocking_user,
5     blocked_activity.query AS blocked_statement,
6     blocking_activity.query AS current_statement_in_blocking_process
7 FROM pg_catalog.pg_locks blocked_locks
8 JOIN pg_catalog.pg_stat_activity blocked_activity ON blocked_activity.pid = blocked
9 _locks.pid
10 JOIN pg_catalog.pg_locks blocking_locks
11 ON blocking_locks.locktype = blocked_locks.locktype
12 AND blocking_locks.database IS NOT DISTINCT FROM blocked_locks.database
13 AND blocking_locks.relation IS NOT DISTINCT FROM blocked_locks.relation
14 AND blocking_locks.page IS NOT DISTINCT FROM blocked_locks.page
15 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
24     JOIN pg_catalog.pg_stat_activity blocking_activity ON blocking_activity.pid = blockin
25     g_locks.pid
26     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_
9          blks_read,0) + coalesce(tidb_lks_read,0)) ) as from_disk,
10          sum( (coalesce(heap_blks_hit,0) + coalesce(idx_blks_hit,0) + coalesce(toast_blk
11          s_hit,0) + coalesce(tidb_lks_hit,0)) ) as from_cache
12  FROM pg_statio_all_tables --> change to pg_statio_USER_tables if you want to c
13  heck only user tables (excluding postgres's own tables)
14  ) a
15  WHERE (from_disk + from_cache) > 0 -- discard tables without hits
16  ),
17  tables as
18  (
19  SELECT *
20  FROM (

```

```

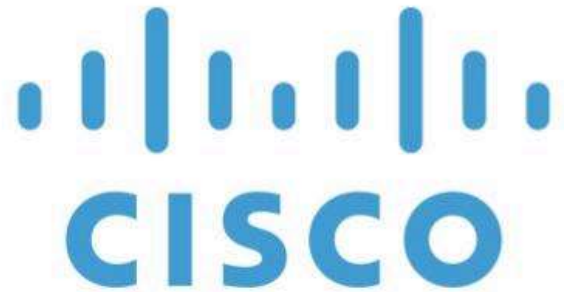
18  SELECT relname as table_name,
19      ( (coalesce(heap_blks_read,0) + coalesce(idx_blks_read,0) + coalesce(toast_blks
      _read,0) + coalesce(tidx_blks_read,0)) ) as from_disk,
20      ( (coalesce(heap_blks_hit,0) + coalesce(idx_blks_hit,0) + coalesce(toast_blks_hi
      t,0) + coalesce(tidx_blks_hit,0)) ) 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**



PostgreSQL Consulting and Support from MinervaDB

- ★ PostgreSQL Performance Benchmarking
- ★ Capacity Planning and Sizing
- ★ Performance Optimization and Tuning
- ★ PostgreSQL High Availability Solutions
- ★ PostgreSQL Replication and Scale-out
- ★ PostgreSQL Database SRE Services
- ★ PostgreSQL Upgrades & Migration
- ★ PostgreSQL Data Recovery
- ★ PostgreSQL Troubleshooting

- ★ Enterprise-Class PostgreSQL Consulting
- ★ PostgreSQL 24*7 Consultative Support
- ★ PostgreSQL Remote DBA Services
- ★ PostgreSQL Emergency Support
- ★ PostgreSQL On-Demand DBA Services
- ★ PostgreSQL Monitoring (24*7)
- ★ Bug Fixing and Patching*
- ★ PostgreSQL Data Security
- ★ PostgreSQL DevOps. / Automation

Email - contact@minervadb.com
Phone (Toll free) - (844) 588-7287
Fax - +1 (209) 314-2364