

# Identificar índices não utilizados (ou não eficientes)

```
with
table_scans as (
select
[]tables.relid as relid,
[]tables.schemaname as schemaname,
[]tables.relname as tablename,
[]tables.idx_scan as table_idx_scan_count,
[]tables.idx_tup_fetch as table_idx_tup_fetch,
[]tables.seq_scan as table_seq_scan_count,
[]tables.seq_tup_read as table_seq_tup_read,
[]tables.idx_scan + tables.seq_scan as table_sum_all_scans,
[]tables.n_tup_ins as table_write_insert_count,
[]tables.n_tup_upd as table_write_update_count,
[]tables.n_tup_del as table_write_delete_count,
[]tables.n_tup_ins + tables.n_tup_upd + tables.n_tup_del as table_sum_all_writes,
[]tables.n_tup_hot_upd as table_tup_hot_upd_count,
[]tables.n_live_tup as table_live_tup_count,
[]pg_relation_size(relid) as table_bytes
from
[]pg_stat_user_tables as tables
),
database_writes as (
select
[]sum(table_sum_all_writes) as database_sum_all_writes
from
[]table_scans
),
indexes as (
select
[]idx_stat.relid as relid,
[]idx_stat.indexrelid as indexrelid,
[]idx_stat.schemaname as schemaname,
```

```

idx_stat.relname as tablename,
idx_stat.indexrelname as indexname,
idx_stat.idx_scan as index_idx_scan_count,
idx_stat.idx_tup_read as index_idx_tup_read,
idx_stat.idx_tup_fetch as index_idx_tup_fetch,
pg_relation_size(idx_stat.indexrelid) as index_bytes,
indexes.indexdef ~* 'USING btree' as idx_is_btree
from
pg_stat_user_indexes as idx_stat
join pg_index as pg_index
  on (idx_stat.indexrelid = pg_index.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
indexes.schemaname as schemaname,
indexes.tablename as tablename,
indexes.indexname as indexname,
indexes.index_idx_scan_count as index_idx_scan_count,
indexes.index_idx_tup_read as index_idx_tup_read,
indexes.index_idx_tup_fetch as index_idx_tup_fetch,
round(case when indexes.index_idx_scan_count = 0 or table_scans.table_live_tup_count = 0
then -1 :: numeric
      else indexes.index_idx_tup_fetch :: numeric / indexes.index_idx_scan_count /
table_scans.table_live_tup_count * 100 end, 2) as idx_pct_table_fetched,
table_scans.table_idx_scan_count as table_idx_scan_count,
table_scans.table_seq_scan_count as table_seq_scan_count,
table_scans.table_seq_tup_read as table_seq_tup_read,
table_scans.table_sum_all_scans as table_sum_all_scans,
round((case when table_scans.table_sum_all_scans = 0
then -1 :: numeric
      else indexes.index_idx_scan_count :: numeric / table_scans.table_sum_all_scans * 100 end),
2) as index_scan_pct,
table_scans.table_write_insert_count as table_write_insert_count,
table_scans.table_write_update_count as table_write_update_count,

```

```

[]table_scans.table_write_delete_count as table_write_delete_count,
[]table_scans.table_sum_all_writes as table_sum_all_writes,
[]round((case when table_scans.table_sum_all_writes = 0
then indexes.index_idx_scan_count :: numeric
      else indexes.index_idx_scan_count :: numeric / table_scans.table_sum_all_writes end), 2)
as scans_per_write,
[]table_scans.table_tup_hot_upd_count as table_tup_hot_upd_count,
[]table_scans.table_live_tup_count as table_live_tup_count,
[]indexes.index_bytes as index_bytes,
[]pg_size_pretty(indexes.index_bytes) as index_size,
[]table_scans.table_bytes as table_bytes,
[]pg_size_pretty(table_scans.table_bytes) as table_size,
[]indexes.idx_is_btree as idx_is_btree
from
[]indexes
join table_scans
[][]using (relid)
),
index_groups as (
select
[]1 as grp,
[] 'Never Used Indexes' as reason,
[]*
from
[]index_ratios
where
[]index_ratios.index_idx_scan_count = 0
[]and index_ratios.idx_is_btree
union all
select
[]2 as grp,
[] 'Low Scans, High Writes' as reason,
[]*
from
[]index_ratios
where
[]scans_per_write <= 1
[]and index_scan_pct < 10
[]and index_idx_scan_count > 0
[]and table_sum_all_writes > 100
[]and idx_is_btree

```

```

union all
select
  3 as grp,
  'Seldom Used Large Indexes' as reason,
  *
from
  index_ratios
where
  index_scan_pct < 5
  and scans_per_write > 1
  and index_idx_scan_count > 0
  and idx_is_btree
  and index_bytes > 100000000
union all
select
  4 as grp,
  'High-Write Large Non-Btree' as reason,
  index_ratios.*
from
  index_ratios,
  database_writes
where
  (table_sum_all_writes :: numeric / coalesce(nullif(database_sum_all_writes, 0), 1)) > 0.02
  and not idx_is_btree
  and index_bytes > 100000000
union all
select
  5 as grp,
  '(+) Sem Efetividade' as reason,
  index_ratios.*
from
  index_ratios
where
  idx_is_btree
  and index_idx_scan_count > 0
  and idx_pct_table_fetched > 20
union all
select
  6 as grp,
  '(+) Índice Médio (100MB a 500MB)' as reason,
  index_ratios.*

```

```

from
  index_ratios
where
  index_bytes >= 1000000000
  and index_bytes < 5000000000
union all
select
  7 as grp,
  (+) Índice Grande (500MB a 1 GB)' as reason,
  index_ratios.*
from
  index_ratios
where
  index_bytes >= 5000000000
  and index_bytes < 10000000000
union all
select
  8 as grp,
  (+) Índice Enorme (mais de 1 GB)' as reason,
  index_ratios.*
from
  index_ratios
where
  index_bytes >= 10000000000
order by
  grp,
  index_bytes desc
)
select
  reason,
  schemaname,
  tablename,
  indexname,
  table_size,
  index_size
from
  index_groups

```

Revisão #3

Criado 22 December 2021 17:38:43 por Nicolly Andrielly

Atualizado 28 January 2025 17:28:37 por Nicolly Andrielly