

Эффективность индексов PostgreSQL

 chatgpt.com/c/680f83aa-31e4-8006-bb7c-2d5dc111fb3f



Отличный запрос. Давай системно пройдемся по всем метрикам, которые ты указал, и я дам перечень формул, поясню их смысл и назначение в контексте **анализа эффективности индексов в PostgreSQL**.

Переменные, которые ты указал:

- `index_size_bytes` — размер всех индексов в байтах.
- `table_size_bytes` — полный размер таблицы (с данными + индексами) в байтах.
- `index_size_percent_of_table` — доля размера индексов относительно таблицы (процент).
- `index_scans` — количество обращений к индексам (индексных сканов).
- `index_scans_percent` — процент запросов, которые используют индекс.
- `tuples_read` — сколько строк просмотрено при сканировании индексов.
- `tuples_read_percent` — процент строк, считанных через индексы.
- `tuples_fetched` — сколько строк реально возвращено через индекс.

- `tuples_fetched_percent` — процент реально полученных строк от всех прочитанных.
 - `live_rows` — количество "живых" строк в таблице.
 - `dead_rows` — количество "мертвых" строк в таблице (непомеченных на `autovacuum`).
 - `dead_rows_percent` — процент мертвых строк относительно всех.
-

Формулы с комментариями:

1. Процент размера индексов относительно таблицы

Формула:

sql

```
index_size_percent_of_table = (index_size_bytes::numeric / table_size_bytes) *  
100
```

Комментарий:

Показывает, сколько "веса" в таблице занимают индексы. Если более 50-60%, возможно, стоит оптимизировать индексацию.

2. Процент использования индексов в запросах

Формула:

sql

```
index_scans_percent = (index_scans::numeric / (index_scans + seq_scans)) * 100
```

(где *seq_scans* — количество последовательных сканов таблицы)

Комментарий:

Чем выше этот процент, тем эффективнее используются индексы. Иначе часто идут полные сканы таблиц (что плохо).

3. Процент прочитанных кортежей через индекс

Формула:

sql

```
tuples_read_percent = (tuples_read::numeric / (tuples_read +  
tuples_returned_seqscan)) * 100
```

(где *tuples_returned_seqscan* — строки, считанные через полные сканы)

Комментарий:

Показывает, через какие механизмы читаются данные — через индексы или скан всей таблицы.

4. Эффективность чтения через индекс

Формула:

sql

```
tuples_fetched_percent = (tuples_fetched::numeric / tuples_read) * 100
```

Комментарий:

Из всех просмотренных через индекс строк — сколько реально понадобилось.
Если процент низкий, возможно, индекс используется неэффективно (много лишних чтений).

5. Процент "мертвых" строк

Формула:

sql

```
dead_rows_percent = (dead_rows::numeric / (live_rows + dead_rows)) * 100
```

Комментарий:

Большое количество dead-строк (>20%) ухудшает эффективность индексов, потому что они вынуждены вести себя как по "грязным" данным. Требуется регулярный **VACUUM/AUTOVACUUM**.

Дополнительные производные формулы:

6. Индексная плотность данных

Формула:

sql

```
index_density = tuples_read::numeric / index_size_bytes
```

Комментарий:

Показывает, сколько кортежей приходится на байт индекса. Низкая плотность → большой индекс с малой полезной отдачей.

7. Избыточность индексов

Формула:

sql

```
index_bloat_estimate = (actual_index_size - expected_index_size)::numeric /  
actual_index_size * 100
```

*(Требует анализа **bloated** индексов через инструменты типа **pgstattuple** или **pg_bloat_check**.)*

Комментарий:

Накопление "мусора" в индексах приводит к разрастанию размеров и снижению скорости поиска.

Что можно сделать на практике?

- Высокий **index_size_percent_of_table** → проверить необходимость всех индексов.

- **Низкий** `index_scans_percent` → проанализировать планы запросов (`EXPLAIN ANALYZE`), возможно, индексы не задействуются.
 - **Низкий** `tuples_fetched_percent` → возможно, индекс плохо подобран (например, слишком широкий).
 - **Высокий** `dead_rows_percent` → усилить политику очистки данных (`VACUUM/AUTOVACUUM`, `HOT-updates`).
 - **Высокий** `index_bloat_estimate` → регулярно выполнять `REINDEX`.
-

Хочешь, я сразу ещё предложу **готовую таблицу-отчет** на SQL, чтобы собрать все эти метрики сразу по всем таблицам базы? 🚀
(если да — скажи, сгенерирую под PostgreSQL 14+)