RLS Performance and Best Practices #14576

**Locked**

[**1**](https://github.com/GaryAustin1)

[**GaryAustin**](https://github.com/GaryAustin1)

announced in

[**Troubleshootin**](https://github.com/orgs/supabase/discussions/categories/troubleshooting)

[**g**](https://github.com/orgs/supabase/discussions/categories/troubleshooting)

**Category**

[🚩](https://github.com/orgs/supabase/discussions/categories/troubleshooting)

[**Troubleshootin**](https://github.com/orgs/supabase/discussions/categories/troubleshooting)

[**g**](https://github.com/orgs/supabase/discussions/categories/troubleshooting)

**Labels**

None yet

**1**

**participant**



**Events**



[**GaryAustin**](https://github.com/GaryAustin1)

[**1**](https://github.com/GaryAustin1)

y

25, 2023

on Ma

**Maintainer**

*This is a copy of a troubleshooting article on Supabase's docs site. It may be missing some details from the original. View the*

[*original articl*](https://supabase.com/docs/guides/troubleshooting/rls-performance-and-best-practices-Z5Jjwv)

[*e*](https://supabase.com/docs/guides/troubleshooting/rls-performance-and-best-practices-Z5Jjwv)

[*.*](https://supabase.com/docs/guides/troubleshooting/rls-performance-and-best-practices-Z5Jjwv)

**RLS performance and best practices**

Although most of the time spent on thinking about RLS is to get it to handle security needs, the impact of it on

performance of your queries can be massive.

This is especially true on queries that look at every row in a table like for many select operations and updates.

Note that queries that use limit and offset will usually have to query all rows to determine order, not just the limit amount

so they are impacted too.

See the last section for ways to measure the performance of your queries as you test RLS improvements.

**Is RLS causing performance issues (on a single table query)?**

For very slow queries, or if using the tools at end of article, run a query with RLS enabled on the table and then with it

disabled (this should only be done in a non production environment). If the results are similar then your query itself is

likely the performance issue. Although, remember any join tables in RLS will also need to run their RLS unless a security

definer function is used to bypass them. You can also create a service\_role client to run the query bypassing RLS if in a

secure environment.

**How to improve RLS performance.**

The following tips are very broad and each may or may not help the specific case involved. Some changes, like adding

indexes, should be backed out if they do not make a difference in RLS performance and you are not using them for

filtering performance.

**1**

**. The first thing to try is put an index on columns used in the RLS that are not primary keys or unique already.**

For RLS like:

auth.uid() = user\_id

Add an index like:

create index userid on test\_table using btree (user\_id) tablespace pg\_default;

Improvement seen over 100x on large tables.

**2**

**. Another method to improve performance is to wrap your RLS queries and functions in select statements.**

This method works well for JWT functions like

auth.uid()

and

auth.jwt()

as well as any other functions including

security definer type.

Wrapping the function in some SQL causes an

initPlan

to be run by the optimizer which allows it to "cache" the results

versus calling the function

on each row.

WARNING: You can only do this if the results of the query or function do not change based on the row data.

For RLS like this:

is\_admin() or auth.uid() = user\_id

Use this instead:

select is\_admin()) OR (select auth.uid()) = user\_id

(

**. Do not rely on RLS for filtering but only for security.**

**3**

Instead of doing this (JS client example):

.from('table').select()

With an RLS policy of:

auth.uid() = user\_id

Add a filter in addition to the RLS:

.from('table').select().eq('user\_id',userId)

**4**

**. Use security definer functions to do queries on other tables to bypass their RLS when possible.**

**edited by supabase-search**

**bot**

[**GaryAustin**](https://github.com/GaryAustin1)

[**1**](https://github.com/GaryAustin1)

Locked

years ago

2

[**e**](https://github.com/supabase)

[**supabas**](https://github.com/supabase)



[w](https://github.com/supabase)

[Overvie](https://github.com/supabase)

[s](https://github.com/orgs/supabase/repositories)

[Repositorie](https://github.com/orgs/supabase/repositories)

[**Discussion**](https://github.com/orgs/supabase/discussions)

[**s**](https://github.com/orgs/supabase/discussions)

[Project](https://github.com/orgs/supabase/projects)

[s](https://github.com/orgs/supabase/projects)

[Package](https://github.com/orgs/supabase/packages)

[s](https://github.com/orgs/supabase/packages)

[Peopl](https://github.com/orgs/supabase/people)

[e](https://github.com/orgs/supabase/people)

[Sponsorin](https://github.com/orgs/supabase/sponsoring)

[g](https://github.com/orgs/supabase/sponsoring)

[**2**](https://github.com/orgs/supabase/sponsoring)

Instead of having this RLS where the roles\_table has an RLS select policy of auth.uid() = user\_id : exists (select 1 from roles\_table where auth.uid() = user\_id and role = 'good\_role') Create a security definer function has\_role() and do:

(select has\_role()) with code of exists (select 1 from roles\_table where auth.uid() = user\_id and role =

'good\_role')

Note that you should wrap your security definer function in select if it is a fixed value per 2.

Remember functions you use in RLS can be called from the API.

Secure your functions in an alternate schema if their results would be a security leak.

Warning: If your security definer function uses row information as an input parameter be sure to test performance as you can't wrap the function as in 2.

**5. Always optimize join queries to compare row columns to fixed join data.**

Instead of querying on a row column in a join table WHERE, organize your query to get all the column values that meet your query into an array or set.

Then use an IN or ANY operation to filter against the row column.

This RLS to allow select only for rows where the team\_id is one the user has access to: auth.uid() in (select user\_id from team\_user where team\_user.team\_id = table.team\_id)

will be much slower than: team\_id in (select team\_id from team\_user where user\_id = auth.uid())

Also consider moving the join query to a security definer function to avoid RLS on join table: team\_id in (select user\_teams())

Note that if the in list gets to be over 10K items, then extra analysis is likely needed. See this follow up testing:

<https://github.com/GaryAustin1/RLS-Performance/tree/main/tests/Supabase-Docs-Test>.

**6. Use role in TO option or roles dropdown in the dashboard.**

Never just use RLS involving auth.uid() or auth.jwt() as your way to rule out 'anon' role.

Always add 'authenticated' to the approved roles instead of nothing or public. Although this does not improve the query performance for the signed in user it does eliminate 'anon' users without taxing the database to process the rest of the RLS.

# Sample results

The code used for the below tests can be found here: [LINK:](https://github.com/GaryAustin1/RLS-Perfomance)

The tests are doing selects on a 100K row table. Some have an additional join table.

Show RLS and before after for above examples.

# Tools to measure performance

Postgres has tools to analyze the performance of queries. <https://www.postgresql.org/docs/current/sql-explain.html>The use of explain in detail for query analysis is beyond the scope of this discussion.

Here we will use it mainly to get a performance metric to compare times.

In order to do RLS testing you need to setup the user JWT claims and change the running user to anon or

This will return results like:

Seq Scan on rlstest (cost=0.00..4334.00 rows=1 width=35) (actual time=170.999..170.999 rows=0 loops=1)

" Filter: ((COALESCE(NULLIF(current\_setting('request.jwt.claim.sub'::text, true), ''::text), ((NULLIF(current\_setting('request.jwt.claims'::text, true), ''::text))::jsonb ->> 'sub'::text)))::uuid = user\_id)"

## 0 comments