

SQL Query Optimizer by LLM

Ruogu Du
07/2023

SQL Query Optimization is a Trouble

Databricks, Snowflake, Microsoft, and Oracle all have big teams dedicated to this field. Meanwhile, insufficient SQL queries are still everywhere and wasting tons of time and computing resources.

Solutions? Nah


Currently, there are various ways to rewrite an inefficient SQL query, there are even startups just to help you write good queries. But their approaches are still mostly by rules and heuristics.

ChatGPT is able to rewrite some queries, but firstly the current results are not good enough, and secondly, it would be hard to incorporate data schema information such as table, index, etc.

Our Solution, Powered by PaLM

Summary of the APIs

```
{
  "Endpoints": {
    "connect_db": "/connect_db",
    "optimize": "/optimize",
    "index": "/index",
    "explain": "/explain"
  },
  "Instructions": {
    "DB_Login": "To connect to the database, use the 'connect_db' endpoint with a login message formatted as follows:\n\nThe database address is [address].\n\nThe port is [port].\n\nThe username is [username].\n\nThe password is [password].\n\nThe db name is [dbname].\n\nReplace the bracketed terms with your database's details. If not provided, port will default to 5432.",
    "Optimize_Query": "After connecting to the database, use the 'optimize' endpoint to optimize your SQL queries. Pass the SQL query as a string. Ensure you are connected to the database first by using the 'connect_db' endpoint.",
    "Find_Index": "Use the 'index' endpoint to find the optimal index for a given SQL query. Pass the SQL query as a string. Ensure you are connected to the database first by using the 'connect_db' endpoint.",
    "Explain_Query": "Use the 'explain' endpoint to explain a SQL query. Pass the SQL query as a string. Ensure you are connected to the database first by using the 'connect_db' endpoint."
  }
}
```

[Download](#)

Login with Natural Language

CLI, no more.

The database address is 34.31.178.164.

The port is 5432.

The username is googlehackathonuser.

The password is googlehackathonpassword!!!newbee.

The db name is tpcc.

Optimize

Get query optimization result with one click.

Parameters Cancel

Name	Description
human_msg * required string <i>(query)</i>	<input type="text" value="SELECT c.c_w_id, c.c_d_id, c.c_id, c.c_di:"/>
token string <i>(cookie)</i>	<input type="text" value="token"/>

ExecuteClear

What about Indexes

Want to know which indexes to add? This was a huge pain to developers and SREs.

It's no longer a problem!

```
{  
  
  "response": "The following indexes would improve the efficiency of the  
query:\n\n* **customer_id_index:** This index would improve the performance  
of queries that filter on the customer_id column.\n\n* **district_id_index:** This  
index would improve the performance of queries that filter on the district_id  
column.\n\n* **history_id_index:** This index would improve the performance of  
queries that filter on the history_id column.\n\n* **oorder_id_index:** This index  
would improve the performance of queries that filter on the oorder_id  
column.\n\n* **order_line_id_index:** This index would improve the  
performance of queries that filter on the order_line_id column.\n\n\nIn addition,  
you could also create a compound index on the following columns:\n\n*  
**customer_id, district_id, history_id:** This index would improve the  
performance of queries that filter on multiple columns of the customer, district,  
and history tables.\n\n* **oorder_id, customer_id, district_id:** This index would  
improve the performance of queries that filter on multiple columns of the  
order, customer, and district tables.\n\n* **order_line_id, oorder_id,  
customer_id, district_id:** This index would improve the performance of  
queries that filter on multiple columns of the order_line, oorder, customer, and  
district tables."  
  
}
```

Get Execution Plan with Ease

Know the execution plan helps you understand how the database run this query internally.

POST /explain Handle Explain

Endpoint to handle explaining a Postgres query. The chat context is loaded from a cookie.

Parameters Cancel

Name	Description
human_msg * required string (query)	<input type="text" value="SELECT c.c_w_id, c.c_d_id, c.c_id, c.c_disc"/>
token string (cookie)	<input type="text" value="token"/>

Execute Clear

Thank you!