**Teradata DBQL Performance Tuning – Part 1- Setup**

By

[Roland Wenzlofsky](https://www.dwhpro.com/author/rwenzlofsky/)

-

https://www.dwhpro.com/teradata-dbql-performance-tuning-part-1-setup/

**The Teradata DBQL Tables**

This is the introduction post of a series of posts about SQL tuning with DBQL tables.

The DBQL Tables are an important tool for the Teradata SQL tuner. The DBQL environment consists of several tables located in database DBC. Each table gives some specific insight into performance characteristics of queries which are executed on the system.

Here is a short overview about the main tables which are used in performance tuning:

**DBQLOGTBL**

DBQLOGTBL contains, for example, the following useful information for each query:

* Query Start Time and Query End Time
* Parsing Time of the Query
* Error Codes
* Total CPU Usage
* Total IO Usage
* Workload Delay Times
* The Query Text (truncated)

Logging in DBQLOGTBL usually is activated on each Teradata System (although filter criteria may be applied, we will talk about this later).

**DBQLSQLTBL**

If logging is activated, this DBQL table will contain the full query text for each query. Sometimes it is necessary to identify our workload by parsing the query text. In this case, DBQLSQLTBL can be used.

**DBQLOBJTBL**

If activated, tracks the usage of all database objects, such as databases, tables, columns, indexes, etc.  
We can, for example,  use this table to identify unused secondary indexes, to check the skew of used tables, etc.

**DBQLSTEPTBL**

This DBQL tables will log each query step (as seen in the Explain Statement). This includes the following information:

* Query Step Timings (Start and End Time of each step)
* Row Counts per Step
* Activity of each Step (Join, Redistribution, etc.)
* CPU and IO usage for each step

This information gives valuable insight into each query and can be considered very important for performance tuning. Unfortunately, logging is often deactivated, as Teradata Admins fear the resource usage overhead created by logging into DBQLSTEPTBL.

**DBQLSUMMARYTBL**

This table if summary or threshold logging is activated. It contains aggregated response times, CPU time and IO usage. Not that useful to optimize single SQL queries.

**DBQLEXPLAINTBL**

This table holds the whole Explain Plan for each query.

**Setting Up Query Logging**

In my experience, by default, query logging should be set up for all objects. As a minimum recommendation I would activate DBQLOGTBL (Main Table), DBQLOBJLOG (The Object Log) and DBQLSQLTBL without any thresholds/filtering:

BEGIN QUERY LOGGING WITH SQL, OBJECTS LIMIT SQLTEXT=0 ON ALL;

Teradata logs this information in a very efficient way and the impact on system performance is negligible.

Logging for users, accounts, and applications which should not be logged in the default way described above, should be changed on demand. Here are two examples:

BEGIN QUERY LOGGING LIMIT THRESHOLD=100 CPUTIME AND SQLTEXT=5000 ALL ACCOUNT = ‘Account’;

BEGIN QUERY LOGGING LIMIT THRESHOLD=100 CPUTIME AND SQLTEXT=5000 ALL ON APPLNAME = ‘AppName’;

I would not recommend approaching this topic the other way around i.e. turning on query logging only on demand. All Teradata developers should have access to the useful information available in the DBQL tables. How else should they write performing statements?

In our next article, we will show you how performance indicators of the DBQL tables can be used to identify product joins, missing indexes, etc.

Further, we will show you some SQL statements you can use to classify your queries into “good” and “bad” ones.