

Teradata Assessment

Query Execution Logs Extraction and Source Code

Prerequisites

Contents

- 1. Introduction3
- 2. Assessment Process.....3
 - 2.1 Logs.....3
 - 2.1.1 Option 1: TPT Export (Recommended)4
 - 2.1.2 Option 2: FastExport (Not Preferred)4
 - 2.2 Total I/O Usage by Days7
 - 2.3 Hardware Configuration.....8
 - 2.4 Database Objects8
 - 2.5 Database Volume8
 - 2.6 High Data Volume Tables8
 - 2.7 Databases and Users9
 - 2.8 Data for Partitioning / Bucketing9
- 3. Database Object DDL Extraction11
- 4. Source Code Assessment.....12

1. Introduction

LeapLogic Assessment profiles existing inventory, identifies complexity, draws out dependency structure and provides comprehensive recommendations for migration to modern data platform.

Assessment checklist

Information	Format	Availability?
Teradata Query Logs (TPT file attached)	File Export in text format	Choose an item.
Total I/O Usage by Days	File Export in text format	Choose an item.
Teradata Hardware Config for Total CPU Calculation	Inline in the document	Choose an item.
Database Object Count	File Export in text format	Choose an item.
Database Volume	File Export in text format	Choose an item.
High Data Volume Tables	File Export in text format	Choose an item.
Total Number of Databases and Users	File Export in text format	Choose an item.
Data for Partitioning bucketing	File Export in text format	Choose an item.
Source Code	File Export	Choose an item.

Follow the steps in section 2, 3 to collect the required information for assessment.

2. Assessment Process

All the environment specific variables are highlighted in the document in **Green**. The start and end date should be consistent for all the queries.

2.1 Logs

This requires query execution logs in a CSV format. The CSV file must have the following columns (~~ separated file) retrieved from Dbqlgtbl and QryLogSQL tables of PDCRInfo/DBC database.

```
FILLER~~UserName~~AppID~~ClientID~~StartTime~~AMPCPUTime~~TotalIOCount~~ParserCPUTime~~FirstRespTime~~FirstStepTime~~ProcID~~QueryID~~MaxAMPCPUTime~~MaxAmplIO~~totalCPU~~totalIO~~Query_Execution_Time~~OBJECTDATABASENAME~~SQLROWNO~~StatementType~~ LogonSource~~QUERYBAND~~SQLTEXTINFO
```

Two options are available for extracting query execution logs: the first involves utilizing the TPT script, while the second involves using FastExport. We recommend to use the TPT script for query log extraction.

Important Note: Option 1, which uses TPT Export, is recommended for extracting query execution logs. Option 2 (FastExport) should not be used unless Options 1 can't be used pertaining to certain reasons. This is because Option 2 also exports special characters which needs special cleansing


exercise before performing assessment. Therefore, it becomes an additional overhead and may increase assessment time and effort.

2.1.1 Option 1: TPT Export (Recommended)

An effective approach to extract information from query logs is by utilizing a TPT script. This method uses the TPT utility to retrieve the query execution logs, resulting in the generation of a log specific to the provided start and end dates.

The typical steps are:

- Copy the following artifact into a folder on the system where TTU (Teradata Tools and Utilities) is installed.

Artifact Name	File
QueryLogExtract.tpt	 QueryLogExtract.tpt

- Next, modify the specified properties in the file.
 - Update the destination path for storing the query log output.

```
VARCHAR DirectoryPath='/home/ec2-user/output',
```

- Update the TPID with the real Teradata system name or IP address.

```
VARCHAR Tdpid='54.234.225.211',
```

- Update the Teradata system's username and password with the real credentials.

```
VARCHAR UserName='dbc',  
VARCHAR UserPAssword='dbc',
```

- Lastly, execute the TPT script using the below command. Please make sure to adjust start and end date for which the query execution log needs to be downloaded or exported.

```
tbuild -f /path/QueryLogExtract.tpt -u "start_date=2023-07-01, end_date=2023-07-20" | tee  
QueryLogExtract.tpt.log
```

2.1.2 Option 2: FastExport (Not Preferred)

An alternative approach to extract information from DBQL logs involves using FastExport. However, it's important to keep in mind that utilizing FastExport may result in larger output file size.

The typical steps are:

- Save the below FastExport script with a filename 'TD_fastexport.fe'

```
.LOGTABLE dbname.fastexp_log;
.LOGON TDPID/username,password;
DATABASE dbname;
.BEGIN EXPORT SESSIONS 10;
.EXPORT OUTFILE D:\path\Teradata_fastExp_query_log.txt
MODE RECORD FORMAT TEXT;
select cast('IDWWM' || '~' ||
    rtrim(cast(tb1.Username as char(128))) || '~' ||
    rtrim(cast(tb1.AppID as char(30))) || '~' ||
    rtrim(cast(tb1.ClientID as char(30))) || '~' ||
    rtrim(cast(tb1.StartTime as char(21))) || '~' ||
    rtrim(cast(tb1.ampcputime as char(10))) || '~' ||
    rtrim(cast(tb1.TotalIOCount as char(20))) || '~' ||
    rtrim(cast(tb1.ParserCPUTime as char(10))) || '~' ||
    rtrim(cast(tb1.firstresptime as char(21))) || '~' ||
    rtrim(cast(tb1.firststeptime as char(21))) || '~' ||
    rtrim(cast(tb1.ProcID as char(6))) || '~' ||
    rtrim(cast(tb1.QueryID as char(20))) || '~' ||
    rtrim(cast(tb1.MaxAMPCPUTime as char(10))) || '~' ||
    rtrim(cast(tb1.MaxAMPIO as char(20))) || '~' ||
    rtrim(cast(tb4.ampcputime as char(10))) || '~' ||
    rtrim(cast(tb4.TotalIOCount as char(20))) || '~' ||
    rtrim(cast(((CAST((CAST(tb1.firstresptime AS DATE)-
CAST(tb1.firststeptime AS DATE)) AS DECIMAL(18,6)) * 60*60*24)
+ ((EXTRACT( HOUR FROM tb1.firstresptime) - EXTRACT( HOUR FROM
tb1.firststeptime)) * 60*60)
+ ((EXTRACT(MINUTE FROM tb1.firstresptime) - EXTRACT(MINUTE FROM
tb1.firststeptime)) * 60)
+ (EXTRACT(SECOND FROM tb1.firstresptime) - EXTRACT(SECOND FROM
tb1.firststeptime)) ) as char(20))) || '~' ||
    rtrim(cast(coalesce(TB3.OBJECTDATABASENAME,'Not Present') as
char(500))) || '~' ||
    rtrim(cast(tb2.sqlrowno as char(2))) || '~' ||
    rtrim(cast(tb1.statementtype as char(100))) || '~' ||
    rtrim(cast(tb1.logonsource as char(100))) || '~' ||
    rtrim(cast(coalesce(tb1.queryband,'Not Present') as char(6160))) || '~' ||
    rtrim(cast(tb2.SQLTEXTINFO as char(20000)))
as char(27000))
from
pdcrinfo.dbqlogtbl tb1 inner join pdcrinfo.dbqsqltbl tb2
```

```

on tb1.queryid = tb2.queryid
and tb1.procid = tb2.procid
and tb1.logdate = tb2.logdate
left outer join
(
select queryid,procid,logdate,tdstats.udfconcat(trim(objectdatabasename)) as
objectdatabasename
from pdcrinfo.dbqlobjtbl
where objecttype='db' and logdate between 2023/01/01 AND 2023/05/29
group by queryid,procid,logdate
)
tb3
on tb1.queryid = tb3.queryid
and tb1.procid = tb3.procid
and tb1.logdate = tb3.logdate
inner join
(
select logdate,sum(ampcputime) ampcputime,sum(totaliocount) as totaliocount
from pdcrinfo.dbqlogtbl
where
logdate between 2023/01/01 AND 2023/05/29
group by logdate
)tb4 on tb4.logdate = tb1.logdate
where
tb1.logdate between 2023/01/01 AND 2023/05/29
and tb1.errorcode=0
order by tb1.procid,tb1.queryid,tb2.sqlrowno;
.END EXPORT;
.LOGOFF;

```

- Highlighted variable to be modified as:
 - Replace 2023/01/01 AND 2023/05/29 with actual export assessment start and end date within the attached query.
 - Replace dbname with actual database name.
 - Replace TDPID with actual Teradata Server IP.
 - Replace username, password with actual database username and password within the attached query.
 - Replace database name dbc with actual database within the attached query.

- Replace DATAFILE path `D:\path\Teradata_fastExp_query_log.txt` with actual path and fileName within the attached query.
- Execute the FastExport script.

```
fexp < TD_fastexport.fe
```

**** Notes for Troubleshooting ****

If the user does not have access to **function tdstats.udfconcat** then:

- Grant the required permission to user before running the script again.

```
GRANT SELECT ON TDSTATS to "USERNAME";
```

- Change `"tdstats.udfconcat(trim(OBJECTDATABASENAME)) as OBJECTDATABASENAME"` to `"trim(OBJECTDATABASENAME) as OBJECTDATABASENAME"`
- Grant CREATE TABLE and DROP TABLE privileges to the Teradata user to create the FastExport log table.

```
GRANT CREATE TABLE, DROP TABLE
ON dbname
TO userid;
```

If the TPT script is modified on a Windows machine and uploaded to a Linux system, then:

- Use **dos2unix** tool to convert text files from DOS line endings (carriage return + line feed) to Unix line endings (line feed).

```
# dos2unix QueryLogExtract.tpt
dos2unix: converting file QueryLogExtract.tpt to UNIX format ...
```

**** Notes for Troubleshooting ****

2.2 Total I/O Usage by Days

Save the results of this query in a CSV file:

```
SELECT THEDATE AS RUNDATAE,
CAST(SUM(FILEACQREADS+FILEPREREADS+FILEWRITES) AS BIGINT) AS TOTALIOREADS
```

```
FROM DBC.RESUSAGESPMA
WHERE THEDATE BETWEEN '2023/01/01' AND '2023/05/29'
GROUP BY THEDATE;
```

Replace '2023/01/01' AND '2023/05/29' with actual export assessment start and end date.

2.3 Hardware Configuration

The details of hardware configuration are required for the calculation of total CPU utilization. Please provide the following details.

- Teradata series and specs (Cores, RAM, HDD/SDD)
- Number of Teradata nodes
- TPerf
- Total available AMPCPU seconds per day

2.4 Database Objects

This query provides the count of database objects in use. Save its results in a CSV file.

```
select databasename,tablekind,count(tablename)
from dbc.tables group by databasename,tablekind order by databasename
```

2.5 Database Volume

This SQL will collect databases with volume equal or above 10GB. Save its results in a CSV file.

```
select databasename,cast (sum(currentperm)/1024/1024/1024 as decimal(18,2))
from dbc.diskspace
group by databasename
having cast (sum(currentperm)/1024/1024/1024 as decimal (18,2)) >= 10 order by 2 desc
```

Note: The filter condition can be changed for collecting lower volumes.

2.6 High Data Volume Tables

This query provides results for tables with high data volume. Save its results in a CSV file.

```
select
ts.databasename,
ts.tablename,
```



```

rc.num_rows,
ts.currentsize_gb,
pv.ConstraintType,
pv.ConstraintText
from
(select
    databasename,tablename,sum(currentperm)/1024/1024/1024 as currentsize_gb
from
    dbc.tablesize
group by 1,2
having sum(currentperm)/1024/1024/1024 >= 10
)ts
left join
(select databasename,tablename,max(sv.rowcount) num_rows from dbc.statstv sv group by
databasename,tablename) rc
on ts.databasename=rc.databasename and ts.tablename=rc.tablename
left join DBC.PartitioningConstraintsv pv
on ts.databasename=pv.databasename and ts.tablename=pv.tablename
order by ts.currentsize_gb desc

```

Note: The filter condition can be changed for collecting lower volumes.

2.7 Databases and Users

This query provides the total number of databases and users. Save its results in a CSV file.

```
select dbkind,count(*) from dbc.databases group by dbkind
```

2.8 Data for Partitioning / Bucketing

This query extracts table size and column details to be utilized for partitioning and bucketing recommendations. Save the results in a CSV file.

```

SELECT cols.databasename      AS DATABASE_NAME,
       cols.tablename         AS Table_Name,
       Cast(alsp.currentperm AS BIGINT) AS TABLE_SIZE_BYTES,
       "                      AS NUM_ROWS,
       cols.columnname        AS Column_Name,
       "                      AS num_unique_val
FROM   dbc.columns AS cols
       JOIN ( select databasename,tablename, cast(sum (currentperm) as bigint) as currentperm from
dbc.allspace group by 1 ,2 ) alsp
       ON cols.tablename = alsp.tablename and cols.databasename = alsp.databasename


```

```
WHERE cols.tablename <> 'All';
```

3. Database Object DDL Extraction

LeapLogic needs specific artifacts to perform an assessment. You can either copy these artifacts from your GIT repository or export them, including table DDL scripts, stored procedures, functions, macros, views, etc., using the shell script provided later in this section.

Please ensure that you run the shell script on a machine where the **TTU (Teradata Tools and Utility)** software is installed.

Shell Script	 teradata_ddl_extrac t.sh
--------------	--

Update the variables in the attached shell script as needed.

```
# Teradata system ID and credentials
tpdtd=172.26.52.161
username=dbc
password=dbc

# list of databases
databases="DS_TBL_DB', 'TESTQLV'"
```

The shell script will create a “scripts” folder containing all the DDLs in the directory where the script is executed.

4. Source Code Assessment

Provide the below active or in-scope Teradata source code artifacts as applicable in the migration scope.

Sl. No.	Code Artifact	Criticality	Remarks
1	Orchestration Scripts (Control-M / Autosys / Cron etc.)	Must	To identify interdependencies across scheduler scripts / jobs, queries, and dependent workloads
2	Procedures / Functions*	Must	To identify workload complexity, query patterns, query count etc. for effort estimation and technical debt
3	Views	Must	To identify view complexity, patterns and effort estimations
4	Shell Scripts*	Must	To identify count, dependencies, SQL queries and PL/SQL, logic (example: email, notification etc.) and effort estimations
5	DDL	Must	To identify column usage, and provide recommendation on column level lineage, and query optimization on the target system
6	DML / SQL Files*	Must	To identify count, dependencies, SQL queries and effort estimations
7	CTL Files*	Must	To identify source data type and formats, example Delimiters, JSON etc.
8	BTEQ*	Must	To identify count, dependencies, SQL queries and PL/SQL, transformation logics and effort estimations
9	TPT, Fexp, mLoad	Must	To identify count, complexity of data ingestion scripts and effort estimation

Note:

Limit: Assuming the orchestration script is a trigger point for every single use case execution in the existing setup. If the customer is not comfortable sharing all the workloads, then share those workloads which are referred to or executed through the orchestration scripts. However, in such scenarios the scope and effort estimates will be based on the given workloads.