

Netezza Assessment Query Execution Logs and Source Code Extraction Prerequisites

Contents

- 1. Introduction3
- 2. Assessment Process3
 - 2.1 Logs.....3
 - 2.2 Total I/O Usage by Days5
 - 2.3 Hardware Configuration.....5
 - 2.4 Database Object Count5
 - 2.5 Database Volume6
 - 2.6 High Data Volume Tables6
 - 2.7 Databases6
 - 2.8 Users.....7
 - 2.9 Data for Partitioning and Bucketing.....7
- 3. Database Object DDL Extraction9
- 4. Source Code Assessment.....10

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?
Netezza Query Logs	File Export in text format	Choose an item.
Total I/O Usage by Days	File Export in text format	Choose an item.
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	File Export in text format	Choose an item.
Total Number of Users	File Export in text format	Choose an item.
Data for Partitioning bucketing	File Export in text format	Choose an item.
Source Code Assessment	File Export	Choose an item.

Follow the steps in Section-2 and Section-3 to extract or export 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 the CSV format. It must have the following columns (~~ separated file) retrieved from the history database tables of Netezza.

```
IDWWM~~NPSID~~NPSINSTANCEID~~OPID~~UserName~~AppId~~ClientId~~StartTime~~AmpCpuTime~~
TotalIoCount~~ParserCpuTime~~FirstRespTime~~FirstStepTime~~ProcID~~QueryID~~MaxAmpCpuTime~~
MaxAmpIO~~TotalCpu~~TotalIO~~QueryExecutionTime~~SchemaName~~SQLROWNO~~StatementType~~
~QueryText
```

Netezza query logs with wrapper script

Script for extracting artifacts from Netezza history database Version 3



`nz_query_log.ksh.tx`
t

Script for extracting artifacts from Netezza history database Version 1



nz_query_log.ksh.tx
t

Download the above file and rename it to **nz_query_log.ksh**.

Follow the steps below for execution of wrapper script for extracting the query logs from Netezza.

1. Login to your Netezza server/ Netezza client machine (Linux-based) where Netezza client utilities are installed.
2. Create a folder and copy the file nz_query_log.ksh inside this folder.
3. Edit the file nz_query_log.ksh for environmental details and date range for which you want to extract the logs as specified below.

Edit the value of the following configuration parameters as per your specific use case.

Path of the file where you want to generate the logs file.

OUT_FILE_PATH='/export/home/nz/nz_hist_log'

Input variables (start date, end date, Netezza db username, Netezza db password)

INPUT_START_DT=2020-04-01

INPUT_END_DT=2020-05-31

NZ_USER=admin

NZ_PASSWORD=password

Provide the comma separated database name for which you want to generate the log file. Leave the value of this parameter blank to generate log files for all the databases.

LOG_ASSESSMENT_DATABASES=NEWTPCDS,TPDS

#Provide the name of history database

HISTDB=HISTDB

#Provide the name of history database user

HISTUSER=HISTDBOWNER

4. Give the execute permission to nz_query_log.ksh file by running the command.

```
chmod +x nz_query_log.ksh
```

5. Run the following command to execute the shell script:

- a. Ensure that the current directory is the above-created folder
- b. Run the command

```
./nz_query_log.ksh
```

6. Execution of preceding step will generate the output file in the form of query logs inside the output folder path that you will specify in the OUT_FILE_PATH parameter in the nz_query_log.ksh file.

Except LOG_ASSESSMENT_DATABASES, setting of all the other mentioned variables is mandatory.

The output would be created in the form of query logs files for the given date range. For example, with the start and end dates in the above Shell script variables, the log file will be generated with the name **nz_out.log.2023-04-01_2023-05-31**

2.2 Total I/O Usage by Days

Save the results of this query in a CSV file:

```
select cast(START_TIME as date) as startDate,  
cast(END_TIME as date) as endDate,  
SUM( HOST_DISK_READ_SECS + HOST_DISK_WRITE_SECS + HOST_FABRIC_SECS + SPU_DISK_READ_SECS  
+ SPU_DISK_WRITE_SECS + SPU_FABRIC_SECS + SPU_DATA_DISK_READ_SECS +  
SPU_DATA_DISK_WRITE_SECS + SPU_TEMP_DISK_READ_SECS + SPU_TEMP_DISK_WRITE_SECS ) as  
TOTALREDWRITE  
from _V_SCHED_GRA_EXT  
group by 1,2  
--limit 10
```

Replace *startDate* and *endDate* with actual export assessment start and end date.

2.3 Hardware Configuration

The hardware configuration is required for the calculation of total CPU utilization.

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

2.4 Database Object Count

Save its results in a CSV file.

```
select database::nvarchar(64) as database,OBJTYPE, count(OBJNAME) as Tables_count
from _V_OBJ_RELATION_XDB
group by database,OBJTYPE;
```

2.5 Database Volume

Save its results in a CSV file.

```
SELECT ORX.database::nvarchar(64) AS "DatabaseName",
case when sum (SOD.allocated_bytes) is null then 0 else SUM(SOD.allocated_bytes)/1073741824 end AS
"AllocatedSpace_GB"
FROM _V_SYS_OBJECT_DSLICE_INFO SOD INNER JOIN _V_OBJ_RELATION_XDB ORX ON ORX.objid =
SOD.tblid
GROUP BY "DatabaseName"
ORDER BY "DatabaseName";
```

2.6 High Data Volume Tables

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

```
select objname as table_name,
       database as db_name,
       (allocated_bytes/1048576) as allocated_mbytes
from _v_sys_relation_xdb sys,
     _v_sys_object_dslice_info ds
where ds.tblid = sys.objid
     and dsid in (1,2,3,4)
     and allocated_mbytes > 10240
order by
allocated_mbytes desc,
table_name,
db_name,
dsid;
```

This SQL will collect databases with volume equal or above 10 GB

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

2.7 Databases

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

```
select distinct database::nvarchar(64)
```

```
from _V_OBJ_RELATION_XDB;
```

2.8 Users

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

```
SELECT GROUPNAME,OWNER,USERNAME
FROM _V_GROUPUSERS;
```

2.9 Data for Partitioning and Bucketing

This query extracts table size and column details to be utilized for partition and bucket recommendation. Save the result in a CSV file.

Please note that this query will give details for the database that we are in. Please run it for all databases. Also, please generate the statistics before executing this query.

```
select DATABASE_NAME,Table_Name,TABLE_SIZE_BYTES,NUM_ROWS,Column_Name,
to_number(num_of_Unique_Values, 99999999999999999999999999999999) as num_unique_val
from(
SELECT
_V_TABLE."DATABASE" as database_name,
_V_TABLE.TABLENAME as Table_Name,
case when tbl_stat.used_bytes is null then 0 else tbl_stat.used_bytes end as table_size_bytes,
_V_TABLE.RELTUPLES as num_rows,
SUBSTR(_v_relation_column.attname || ' ', 1, 25) as Column_Name,
case when _v_relation_column.attdispersion = '0' then '0'
when _v_relation_column.attdispersion = '-1' then _V_TABLE.RELTUPLES::varchar(100)
else
TO_CHAR( (CAST((1.0/_v_relation_column.attdispersion) AS BIGINT)), ' 999,999,999,999,999 ' ) end as
num_of_Unique_Values
FROM _v_relation_column
left outer join _v_statistic on
(_v_relation_column.objid = _v_statistic.objid AND
_v_relation_column.attnum = _v_statistic.attnum
)
inner join _V_TABLE
on (_v_relation_column.objid = _V_TABLE.OBJID)
inner join _v_table_storage_stat as tbl_stat
on(tbl_stat.OBJID = _V_TABLE.OBJID)
WHERE
_V_TABLE.OBJTYPE = 'TABLE'
AND
( _v_relation_column.schema=current_schema OR upper(_v_relation_column.schema) in
('DEFINITION_SCHEMA', 'INFORMATION_SCHEMA'))
```

```
ORDER BY
_v_TABLE.TABLENAME ,
_v_relation_column.attnum
) as a;
```


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.

The shell script depends on the Netezza-supplied admin script **nz_ddl**, which is usually found in the **/nz/support/bin** directory.

```
#!/bin/bash

#Database connection parameters
export NZ_HOST="192.168.218.19"
export NZ_DATABASE="NEWTPCDS"
export NZ_USER="admin"
export NZ_PASSWORD="password"

# Netezza admin Script folder
NZ_ADMIN_SCRIPTS="/nz/support/bin"

# Output DIRECTORY
OUTPUT_DIR="/export/home/nz/ddlExtractor/ddls"

# A comma-separated list of databases from which DDLs should be extracted.
DATABASE_LIST="NEWTPCDS,SYSTEM_BKP,AATESTQLV1,AATESTQLV2"

IFS=',' read -r -a databases <<< "$DATABASE_LIST"

if [ -z "$DATABASE_LIST" ]; then
    sh $NZ_ADMIN_SCRIPTS/nz_ddl > $OUTPUT_DIR/"all_ddl_output.sql"
elif [ ${#databases[@]} -eq 1 ]; then
    DATABASE=${databases[0]}
    sh $NZ_ADMIN_SCRIPTS/nz_ddl $DATABASE > $OUTPUT_DIR/"${DATABASE}_ddl_output.sql"
else
    for DATABASE in "${databases[@]}"; do
        sh $NZ_ADMIN_SCRIPTS/nz_ddl $DATABASE > $OUTPUT_DIR/"${DATABASE}_ddl_output.sql"
    done
fi
```

Update the **highlighted** details in the script and run it on the machine where the **nz_ddl** admin script is available.

Also attached script for reference:



nz_export_ddl.sh

4. Source Code Assessment

Provide the below active or in-scope Netezza 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

Note:

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