République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
ECOLE SUPÉRIEURE EN INFORMATIQUE
8 Mai 1945 - Sidi-Bel-Abbès

الجمهورية الجزائرية الديمقراطية الشعبية
وزارة التعليم العالي والبحث العلمي
المدرسة العليا للإعلام الآلي
8 ماي 1945 - سيدي بلعباس

Oracle Homework Rapport

SQL Developer Access Advisor

Made by: Dada Yacine G2

Introduction

This document provides a comprehensive guide on how to perform a SQL Access Advisor analysis using Oracle SQL Developer. The SQL Access Advisor is a powerful tool that helps optimize database performance by analyzing SQL workloads and recommending improvements. The steps outlined in this guide cover everything from granting necessary privileges to executing the advisor task and viewing the recommendations.

Objective

The objective of this guide is to streamline the process of SQL optimization for database administrators. By following the steps provided, users will be able to:

- Grant the required roles and privileges to the user.
- Import data and execute necessary scripts.
- Create and manage SQL Tuning Sets.
- Capture SQL statements from the Automatic Workload Repository (AWR).
- Create and configure a SQL Access Advisor task.
- View and interpret the results and recommendations provided by the SQL Access Advisor.

Steps to Perform SQL Access Advisor Analysis

1. Grant Required Roles and Privileges

First, ensure that the user TPUSERNEW has the necessary roles and privileges to perform SQL Access Advisor tasks.

```sql
-- Grant necessary roles to the user
GRANT DBA TO TPUSERNEW;
GRANT SYSDBA TO TPUSERNEW;
GRANT ADVISOR TO TPUSERNEW;

-- Verify the roles granted to the user
SELECT * FROM DBA_ROLE_PRIVS WHERE GRANTEE = 'TPUSERNEW';
```

2. Import Data and Execute Script

Import your data tables into the database and execute the script containing the 113 queries.

3. Create a SQL Tuning Set for SQL Access Advisor

Grant the required privileges to TPUSERNEW to create and manage SQL Tuning Sets.

```sql
-- Grant necessary privileges to the user
GRANT ADMINISTER SQL TUNING SET TO TPUSERNEW;
GRANT ADMINISTER ANY SQL TUNING SET TO TPUSERNEW;
```

4. Capture SQL Statements from the Automatic Workload Repository

Create a manual snapshot of the database performance data in the AWR and create a new SQL Tuning Set (STS).

```sql
-- Create a manual snapshot
EXEC DBMS_WORKLOAD_REPOSITORY.CREATE_SNAPSHOT;

-- Optionally, view the SQL text captured in the AWR
SELECT * FROM DBA_HIST_SQLTEXT;

-- Create a new SQL Tuning Set
BEGIN
  DBMS_SQLSET.CREATE_SQLSET (
    sqlset_name  => 'STS_AWR_FOURTH',
    description  => 'STS to store SQL from the private SQL area'
  );
END;
/

-- Verify the created SQL Tuning Set
SELECT NAME, STATEMENT_COUNT AS "SQLCNT", DESCRIPTION
FROM USER_SQLSET;
```

5. Populate SQL Tuning Set with AWR Data

Retrieve the snapshot IDs, populate the SQL Tuning Set with SQL statements from the AWR snapshots, and verify the SQL statements.

SNAP_ID	BEGIN_SNAP	END_SNAP
63	2024-06-10 01:50	2024-06-10 02:00
64	2024-06-10 02:00	2024-06-10 03:00
65	2024-06-10 03:00	2024-06-10 03:39

```sql
-- Set the timestamp format for better readability
ALTER SESSION SET NLS_TIMESTAMP_FORMAT = 'YYYY-MM-DD HH:MI';

-- Retrieve the snapshot IDs and their corresponding begin and end times
COL BEGIN_SNAP FORMAT A20
COL END_SNAP FORMAT A20
SELECT SNAP_ID, BEGIN_INTERVAL_TIME BEGIN_SNAP, END_INTERVAL_TIME END_SNAP
FROM DBA_HIST_SNAPSHOT
ORDER BY SNAP_ID;

-- Populate the SQL Tuning Set with SQL statements from the AWR snapshots
DECLARE
  cur DBMS_SQLTUNE.SQLSET_CURSOR;
BEGIN
  OPEN cur FOR
    SELECT VALUE(P)
    FROM DBMS_SQLTUNE.SELECT_WORKLOAD_REPOSITORY(&begin_snap_id, &end_snap_id, NULL, NULL,
  DBMS_SQLTUNE.LOAD_SQLSET(sqlset_name => 'STS_AWR_FOURTH', populate_cursor => cur);
END;
/

-- Verify the SQL statements in the SQL Tuning Set
COLUMN SQL_TEXT FORMAT A30
COLUMN SCH FORMAT A3
       ELAPSED_TIME AS "ELAPSED", BUFFER_GETS
FROM TABLE(DBMS_SQLTUNE.SELECT_SQLSET('STS_AWR_FOURTH'))
WHERE PARSING_SCHEMA_NAME = 'TPUSERNEW';
```

6. Create and Configure the SQL Access Advisor Task

Create a new SQL Access Advisor task, set its parameters, and add the SQL Tuning Set as a reference for the task.

```sql
-- Declare variables for the task ID and task name
VARIABLE task_id NUMBER;
VARIABLE task_name VARCHAR2(255);

-- Create a new SQL Access Advisor task
EXEC :task_name := 'STS_AWR_FOURTH_TASK';
EXEC DBMS_ADVISOR.CREATE_TASK('SQL Access Advisor', :task_id, :task_name);

-- Set the analysis scope for the task
EXEC DBMS_ADVISOR.SET_TASK_PARAMETER(:task_name, 'ANALYSIS_SCOPE', 'ALL');

-- Add the SQL Tuning Set as a reference for the task
EXECUTE DBMS_ADVISOR.ADD_STS_REF(:task_name, 'TPUSERNEW', 'STS_AWR_FOURTH');

-- Execute the SQL Access Advisor task
EXECUTE DBMS_ADVISOR.EXECUTE_TASK(:task_name);
```

7. View SQL Access Advisor Task Results

Check the task status, view the tasks, recommendations, and actions, and commit the changes.

```sql
-- Check the task status and status message
COL TASK_ID FORMAT 999
COL TASK_NAME FORMAT A25
COL STATUS_MESSAGE FORMAT A25
SELECT TASK_ID, TASK_NAME, STATUS, STATUS_MESSAGE
FROM USER_ADVISOR_LOG;

-- View the tasks
SELECT * FROM USER_ADVISOR_TASKS;

-- View the recommendations for the task
SELECT * FROM USER_ADVISOR_RECOMMENDATIONS WHERE TASK_ID = :task_id;

-- View the actions for the recommendations
SELECT * FROM USER_ADVISOR_ACTIONS WHERE TASK_ID = :task_id;

-- Commit the changes
COMMIT;
```

```
TASK_ID TASK_NAME                 STATUS      STATUS_MESSAGE
------- ------------------------- ----------- -------------------------
     77 staName33410              COMPLETED
     91 STS_AWR_FOURTH_TASK       COMPLETED   Access advisor execution
                                              completed
```

8. Display Recommendations

Execute the provided show_recm procedure to display the recommendations and their attributes.

sql Copy code

```sql
CREATE OR REPLACE PROCEDURE show_recm (in_task_name IN VARCHAR2) IS
  CURSOR curs IS
    SELECT DISTINCT action_id, command, attr1, attr2, attr3, attr4
    FROM user_advisor_actions
    WHERE task_name = in_task_name
    ORDER BY action_id;
  v_action NUMBER;
  v_command VARCHAR2(32);
  v_attr1 VARCHAR2(4000);
  v_attr2 VARCHAR2(4000);
  v_attr3 VARCHAR2(4000);
  v_attr4 VARCHAR2(4000);
  v_attr5 VARCHAR2(4000);
BEGIN
  OPEN curs;
  DBMS_OUTPUT.PUT_LINE('======================================');
  DBMS_OUTPUT.PUT_LINE('Task_name = ' || in_task_name);
  LOOP
    FETCH curs INTO
      v_action, v_command, v_attr1, v_attr2, v_attr3, v_attr4;
    EXIT WHEN curs%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE('Action ID: ' || v_action);
    DBMS_OUTPUT.PUT_LINE('Command : ' || v_command);
    DBMS_OUTPUT.PUT_LINE('Attr1 (name) : ' || SUBSTR(v_attr1, 1, 30));
    DBMS_OUTPUT.PUT_LINE('Attr2 (tablespace): ' || SUBSTR(v_attr2, 1, 30));
    DBMS_OUTPUT.PUT_LINE('Attr3 : ' || SUBSTR(v_attr3, 1, 30));
    DBMS_OUTPUT.PUT_LINE('Attr4 : ' || v_attr4);
    DBMS_OUTPUT.PUT_LINE('Attr5 : ' || v_attr5);
    DBMS_OUTPUT.PUT_LINE('--------------------------------------');
  END LOOP;
  CLOSE curs;
END show_recm;
/
```

	TASK_ID	TASK_NAME	EXECUTION_NAME	REC_ID	ACTION_ID	OBJECT_ID	COMMAND	COMMAND_ID	FLAGS	ATTR1	
1	91	STS_AWR_FOURTH_TASK	(null)	7	51	0	CREATE MATERIALIZED VIEW	3	540728	"TPUSER"."MV$$_005B0002"	(n
2	91	STS_AWR_FOURTH_TASK	(null)	11	53	0	CREATE MATERIALIZED VIEW	3	524344	"TPUSER"."MV$$_005B0003"	(n
3	91	STS_AWR_FOURTH_TASK	(null)	13	55	0	CREATE MATERIALIZED VIEW	3	524344	"TPUSER"."MV$$_005B0004"	(n
4	91	STS_AWR_FOURTH_TASK	(null)	15	47	0	CREATE MATERIALIZED VIEW	3	524344	"TPUSER"."MV$$_005B0000"	(n
5	91	STS_AWR_FOURTH_TASK	(null)	16	49	0	CREATE MATERIALIZED VIEW	3	524344	"TPUSER"."MV$$_005B0001"	(n
6	91	STS_AWR_FOURTH_TASK	(null)	24	57	0	CREATE MATERIALIZED VIEW	3	540680	"TPUSER"."MV$$_005B0005"	(n
7	91	STS_AWR_FOURTH_TASK	(null)	25	59	0	CREATE MATERIALIZED VIEW	3	524344	"TPUSER"."MV$$_005B0006"	(n
8	91	STS_AWR_FOURTH_TASK	(null)	26	57	0	CREATE MATERIALIZED VIEW	3	540680	"TPUSER"."MV$$_005B0005"	(n
9	91	STS_AWR_FOURTH_TASK	(null)	28	61	0	CREATE MATERIALIZED VIEW	3	524344	"TPUSER"."MV$$_005B0007"	(n

Conclusion

By following this comprehensive guide, you can effectively utilize the SQL Access Advisor to analyze and optimize SQL workloads in your Oracle database.The steps ensure that the necessary privileges are granted, data is correctly imported and managed, SQL Tuning Sets are created and populated, and the SQL Access Advisor task is configured and executed.

The recommendations provided by the SQL Access Advisor will help in enhancing database performance, ensuring efficient and effective database operations.