نحوه استفاده از

Oracle SQL Performance Analyzer

امیررضا رستنده

Amirreza.Rastandeh@gmail.com

گاهی اوقات میخواهیم تاثیر تغییرات و اصلاحات انجام شده مثل تغییرات سرور، ارتقا دیتابیس، تغییرات پارامترهای دیتابیس، اضافه یا کم شدن ایندکس، تغییر در پارامترها و ساختار جداول و ... را بر عملکرد یک یا چند کوئری مشاهده کنیم و نمیخواهیم این تغییرات تا زمان بررسی دقیق و تایید قابل قبول بودن آنها در دیتابیس اصلی انجام شوند. از طرف دیگر شرایط اجرا دستور در محیط های مختلف متفاوت است. برای آنکه یک یا چند دستور را با آمار و شرایط اجرای خودش در محیط عملیاتی به محیط دیگری ببریم میتوانیم از SQL Tuning Set ها استفاده کنیم که شامل اطلاعات زیاد و مهمی مثل execution plan، آمارهای زمان اجرا (مثل elapsed time، CPU time، تعداد سطرهای پردازش شده، تعداد سطرهای fetch شده، میزان خواندن از دیسک)، bind variable ها، اطلاعات اسکیما و سایر اطلاعات مهمی است که اوراکل در مورد یک کوئری جمع آوری کرده و از آن استفاده میکند.

در این مثال میخواهیم تاثیر یک تغییر کوچک (ایجاد ایندکس) را بر عملکرد یک دستور بررسی کنیم. پس در مرحله اول برای آنکه بتوانیم با اطلاعات صحیح کار کنیم یک SQL Tuning Set را برای SQL ID مورد نظر در محیط عملیاتی (محیط مبدا) می سازیم که در این مثال نام آن ARA_STS_20220220 است:

```
begin
  dbms_sqltune.create_sqlset('ARA_STS_20220220');
end;


SELECT NAME, STATEMENT_COUNT AS "SQLCNT", DESCRIPTION FROM DBA_SQLSET
where name = 'ARA_STS_20220220';


declare
  cur sys_refcursor;
begin
  open cur for
    select value(p)
      from table(dbms_sqltune.select_cursor_cache('sql_id=''&SQLID''')) p;

  dbms_sqltune.load_sqlset('ARA_STS_20220220', cur);
  close cur;
end;


select *
  from table(dbms_sqltune.select_sqlset(sqlset_name    => 'ARA_STS_20220220',
                                        attribute_list => 'BASIC'));


begin
  dbms_sqltune.create_stgtab_sqlset(table_name      => 'STS_STG_TAB',
                                    schema_name     => 'ARA',
                                    tablespace_name => 'DATA_ARA');
end;


SELECT * FROM ARA.STS_STG_TAB;
```

```
begin
  dbms_sqltune.pack_stgtab_sqlset(sqlset_name          => '&STS_NAME',
                                  sqlset_owner         => 'SYS',
                                  staging_table_name   => 'STS_STG_TAB',
                                  staging_schema_owner => 'ARA');
end;


SELECT * FROM ARA.STS_STG_TAB;
```

توجه داشته باشید که اطلاعات مورد نظر ما در جدول stage با نام ARA.STS_STG_TAB (محیط عملیاتی) قرار دارد.

در محیط تست (محیط مقصد) هم لازم است تا ساختار جدول stage وجود داشته باشد، پس آن را میسازیم:

```
begin
  dbms_sqltune.create_stgtab_sqlset(table_name      => 'STS_STG_TAB',
                                    schema_name     => 'ARA',
                                    tablespace_name => 'DATA_ARA');
end;


SELECT * FROM ARA.STS_STG_TAB;
```

حالا لازم است تا اطلاعات موجود در جدول stage محیط عملیاتی را در جدول stage محیط تست بریزیم، ساده ترین روش استفاده از دیتابیس لینک است:

```
INSERT INTO ARA.STS_STG_TAB@PROD2TEST
SELECT * FROM ARA.STS_STG_TAB;

COMMIT;
```

در محیط تست (مقصد) SQL Tuning Set را لود میکنیم و یک SQL Performance Analyzer Task را ایجاد و آن را اجرا میکنیم:

```
begin
  dbms_sqltune.unpack_stgtab_sqlset(sqlset_name          => '&STS_NAME',
                                    sqlset_owner         => 'SYS',
                                    replace              => true,
                                    staging_table_name   => 'STS_STG_TAB',
                                    staging_schema_owner => 'ARA');
end;


select *  from dba_sqlset_statements s
where s.sqlset_name = '&STS_NAME';
```

```
declare
  lv_task_name varchar2(50);
begin
  lv_task_name:= dbms_sqlpa.create_analysis_task(sqlset_name =>
'ARA_STS_20220220', task_name => 'ARA_SPA_01');
  dbms_output.put_line(lv_task_name);
end;


select task_name, status from user_advisor_tasks
where task_name = 'ARA_SPA_01';


begin
  dbms_sqlpa.execute_analysis_task(task_name        => 'ARA_SPA_01',
                                   execution_type   => 'TEST EXECUTE',
                                   execution_name   => 'MY_SPA_BEFORE');
end;

select task_name, status from dba_advisor_tasks
where task_name = 'ARA_SPA_01';


select dbms_sqlpa.report_analysis_task(task_name => 'ARA_SPA_01',
                                       type      => 'TEXT',
                                       section   => 'SUMMARY') as "Execution
Report"
  from dual;
```

حالا نوبت اجرای تغییرات مورد نظر است. در این مثال به جدول یک ایندکس اضافه کردم که نتیجه ایجاد آن را در گزارش نهایی قابل مشاهده است.

پس از اجرای تغییرات، مجدد Task مربوط به SQL Performance Analyzer را اجرا میکنیم:

```
begin
  dbms_sqlpa.execute_analysis_task(task_name        => 'ARA_SPA_01',
                                   execution_type   => 'TEST EXECUTE',
                                   execution_name   => 'MY_SPA_AFTER');
end;


select dbms_sqlpa.report_analysis_task(task_name => 'ARA_SPA_01',
                                       type      => 'TEXT',
                                       section   => 'SUMMARY') as "Execution
Report"
  from dual;
```

با داشتن اطلاعات قبل و بعد از اجرا، میتوانیم یک مقایسه بین performance قبل و بعد از تغییرات انجام دهیم و یک گزارش از این مقایسه داشته باشیم:

```
begin
  dbms_sqlpa.execute_analysis_task(task_name      => 'ARA_SPA_01',
                                   execution_type => 'COMPARE PERFORMANCE');
end;


select dbms_sqlpa.report_analysis_task(task_name => 'ARA_SPA_01',
                                       type      => 'HTML',
                                       section   => 'ALL') as "Execution
Report"
  from dual;
```

نمونه گزارش تولید شده بدین شکل است:

General Information

Task Information:

Task Name : ARA_SPA_01
Task Owner : SYS
Description :

Workload Information:

SQL Tuning Set Name : ARA_STS_20220220
SQL Tuning Set Owner : SYS
Total SQL Statement Count : 1

Execution Information:

Execution Name	: EXEC_64307	**Started**	: 02/20/2022 11:41:24
Execution Type	: COMPARE PERFORMANCE	**Last Updated**	: 02/20/2022 11:41:24
Description	:	**Global Time Limit**	: UNLIMITED
Scope	: COMPREHENSIVE	**Per-SQL Time Limit**	: UNUSED
Status	: COMPLETED	**Number of Errors**	: 0

Analysis Information:

Before Change Execution:		**After Change Execution:**	
Execution Name	: MY_SPA_BEFORE	**Execution Name**	: MY_SPA_AFTER
Execution Type	: TEST EXECUTE	**Execution Type**	: TEST EXECUTE

Scope	: COMPREHENSIVE	**Scope**	: COMPREHENSIVE
Status	: COMPLETED	**Status**	: COMPLETED
Started	: 02/20/2022 11:39:23	**Started**	: 02/20/2022 11:40:42
Last Updated	: 02/20/2022 11:39:23	**Last Updated**	: 02/20/2022 11:40:42
Global Time Limit	: UNLIMITED	**Global Time Limit**	: UNLIMITED
Per-SQL Time Limit	: UNUSED	**Per-SQL Time Limit**	: UNUSED
Number of Errors	: 0	**Number of Errors**	: 0

Comparison Metric: ELAPSED_TIME

Workload Impact Threshold: 1%

SQL Impact Threshold: 1%

Report Summary

Projected Workload Change Impact:

Overall Impact	: **96.92%**
Improvement Impact	: **96.92%**
Regression Impact	: **0%**

SQL Statement Count

SQL Category	SQL Count	Plan Change Count
Overall	1	1
Improved	1	1

Top 1 SQL Sorted by Absolute Value of Change Impact on the Workload

object_id	sql_id	Impact on Workload	Execution Frequency	Metric Before	Metric After	Impact on SQL	Plan Change
6	5zga84nvq6rv0	96.92%	1	1167	36	96.92%	y

Note: time statistics are displayed in microseconds

Report Details

SQL Details:

Object ID	: 6
Schema Name	: SYS
Container Name	: Unknown (con_dbid: 822309321)
SQL ID	: 5zga84nvq6rv0
Execution Frequency	: 1
SQL Text	: SELECT /*+IM HERE*/* FROM TEST_TBL WHERE OBJECT_NAME = :"SYS_B_0"

Execution Statistics:

Stat Name	Impact on Workload	Value Before	Value After	Impact on SQL
elapsed_time	96.92%	.001167	.000036	96.92%
parse_time	-106.64%	.00277	.005724	-106.64%
cpu_time	100%	.001224	0	100%
user_io_time	0%	0	0	0%
buffer_gets	99.31%	433	3	99.31%
cost	98.28%	116	2	98.28%
reads	0%	0	0	0%
writes	0%	0	0	0%
io_interconnect_bytes	0%	0	0	0%
rows		1	1	

Note: time statistics are displayed in seconds

Notes:

Before Change:

1. The statement was first executed to warm the buffer cache.
2. Statistics shown were averaged over next 9 executions.

After Change:

1. The statement was first executed to warm the buffer cache.
2. Statistics shown were averaged over next 9 executions.

Findings (2):

1. The performance of this SQL has improved.
2. The structure of the SQL execution plan has changed.

Execution Plan Before Change:

Plan Id : 1076567
Plan Hash Value : 602094504

Id	Operation	Name	Rows	Bytes	Cost	Time
0	SELECT STATEMENT		5	2405	116	00:00:01
* 1	TABLE ACCESS FULL	TEST_TBL	5	2405	116	00:00:01

Predicate Information (identified by operation id):

- 1 - filter("OBJECT_NAME"=:SYS_B_0)

Notes

- Dynamic sampling used for this statement (level = 2)

Execution Plan After Change:

Plan Id : 1076568
Plan Hash Value : 2700620191

Id	Operation	Name	Rows	Bytes	Cost	Time
0	SELECT STATEMENT		1	481	2	00:00:01
1	TABLE ACCESS BY INDEX ROWID BATCHED	TEST_TBL	1	481	2	00:00:01
* 2	INDEX RANGE SCAN	TEST_IDX	1		1	00:00:01

Predicate Information (identified by operation id):

- 2 - access("OBJECT_NAME"=:SYS_B_0)

Notes

- Dynamic sampling used for this statement (level = 2)

برای پاک سازی محیط تست از دستورات زیر استفاده میکنیم:

```
begin
  dbms_sqlpa.drop_analysis_task(task_name => 'ARA_SPA_01');
end;


begin
  dbms_sqltune.drop_sqlset(sqlset_owner => 'SYS',sqlset_name =>
'ARA_STS_20220220');
end;


DROP TABLE bpmsdb.STS_STG_TAB PURGE;
```

و برای پاک سازی محیط عملیاتی نیز از دستورات زیر استفاده میکنیم:

```
begin
  dbms_sqltune.drop_sqlset(sqlset_owner => 'SYS',sqlset_name =>
'ARA_STS_20220220');
end;


DROP TABLE bpmsdb.STS_STG_TAB PURGE;
```