SQL PLAN MANAGEMENT 활용

㈜엑셈 컨설팅본부/DB컨설팅팀 장 정민

개요

오라클은 비롯한 많은 관계형 DBMS 에서는 사용자의 SQL 질의를 효율적으로 처리하기 위해 옵티마이저를 사용하고 있다. 옵티마이저는 유저가 수행하는 SQL 을 받아 실행계획을 생성하고, 실제 SQL 은 이 실행계획을 통해서 수행된다. 데이터 베이스 운영 시 평소 잘 수행되던 SQL 이 성능 이슈를 발생 시키는 때가 있는데, 그 원인이 SQL 실행 계획 변화에 있는 경우가 많다.

SQL 의 실행 계획이 변하는 이유는 다양한데, 통계정보의 변경이나 인덱스의 상태 변화, DB 의 파라메터나 version 변경 등에 의해 SQL 의 실행 계획이 변할 수 있다. 그런데 SQL 의 실행계획 변화를 막기 위해 이런 DB 작업을 하지 않을 수는 없다.

이러한 실행 계획의 변화로 인해 발생될지 모르는 성능저하를 예방하기 위해 Oracle 11g 부터 SPM(SQL Plan Management)라는 기능을 제공하고, 이를 이용해 SQL 외부적인 요소에 의한 영향을 최소화 할 수 있다.

SPM의 사용 목적과 특성

SPM 은 SQL 외부적인 요소의 변화에 의한 영향을 최소화 하는데 그 목적이 있다. SPM 의 주요 특성으로는 다음의 두 가지를 들 수가 있다.

1. Execution Plan 의 변경에 의한 성능저하를 사전 예방
2. Plan History 관리를 통한 SQL Plan 의 이력 관리 가능

SPM 은 SQL Plan Baseline(Plan 과 Hint)을 DB 에 저장해 놓고 검증된 실행계획만을 사용할 수 있도록 하면서, 자동으로 변경되는 SQL 의 실행계획을 관리한다. 새로운 실행 계획이 생성될 경우, 검증이 끝날 때까지 사용하지 않도록 하여 SQL 실행계획 변경으로 발생할 수 있는 성능

문제를 사전에 예방할 수 있다. 또 새로 생성된 실행계획은 검증 과정에서 현재의 실행계획과 비교해 성능이 향상된 경우에만 사용할 수 있도록 하는 것이 가능하다.

SPM 사용 순서

SPM 의 사용은 다음과 같은 순서로 진행한다.

- SQL_PLAN_BASELINES 관련 파라미터 변경
- SQL_PLAN_BASELINES 에 실행계획 등록
- DBA_SQL_PLAN_BASELINES 뷰를 통해 확인
- SQL_PLAN_BASELINES 속성 변경을 통한 사용 실행계획 제어

SQL_PLAN_BASELINES 관련 파라메터 변경

관련 파라메터 조회

```
SQL> show parameter sql_plan_baselines
NAME                                 TYPE      VALUE
------------------------------------ ----------- ------------------------------
optimizer_capture_sql_plan_baselines   boolean    FALSE
optimizer_use_sql_plan_baselines       boolean    TRUE
```

- optimizer_use_sql_plan_baselines : SPB 를 활성화 하는 파라메터
- optimizer_capture_sql_plan_baselines : 2 회 이상 수행되는 SQL 을 SPB 에 자동 등록하도록 하는 파라메터

SQL_PLAN_BASELINES에 실행계획 등록&삭제

SPB 에 실행계획을 등록하는 방법은 두 가지가 있는데 DBMS_SPM 패키지를 사용하여 수동으로 직접 등록하는 방법과, optimizer_capture_sql_plan_baselines 파라메터 설정을 통해 자동으로 실행계획을 등록시키는 방법이 있다.

■ **SPB 수동 등록**

SPB Baseline 등록 패키지 내용

```
DBMS_SPM.LOAD_PLANS_FROM_CURSOR_CACHE (
   sql_id            IN  VARCHAR2,
   plan_hash_value   IN  NUMBER   := NULL,
   fixed             IN  VARCHAR2 := 'NO',
   enabled           IN  VARCHAR2 := 'YES') RETURN PLS_INTEGER;
```

■ **등록하려는 SQL 정보 확인**

```
SELECT sql_id ,
       plan_hash_value ,
       sql_fulltext
FROM   v$sql
WHERE  sql_text LIKE '%spmtest%'

SQL_ID        PLAN_HASH_VALUE SQL_FULLTEXT
------------- --------------- ---------------------------------------
93zny6hkjj6s7      1690735414 SELECT * FROM   spmtest WHERE  lv = 10
93zny6hkjj6s7      4164974113 SELECT * FROM   spmtest WHERE  lv = 10
```

■ **SQL_ID 와 PLAN_HASH_VALUE 이용하여 SPB 등록**

```
DECLARE
 pls pls_integer ;
BEGIN
 pls := DBMS_SPM.LOAD_PLANS_FROM_CURSOR_CACHE ( '93zny6hkjj6s7' , '1690735414') ;
 dbms_output.put_line( pls || '개 등록' ) ;
END ;
/
```

SQL_ID 와 PLAN_HASH_VALUE 를 실행계획을 유일하게 식별 가능하다. 만약 PLAN_HASH_VALUE 값을 입력하지 않을 경우 SQL_ID 에 해당하는 PLAN 을 모두 SPB 에 등록한다.
optimizer_capture_sql_plan_baselines 파라메터를 true 로 설정하면 2 회 이상 수행되는 모든 SQL 의 PLAN 을 자동으로 SPB 에 등록한다.

■ SPB Baseline 삭제 패키지

```
DBMS_SPM.DROP_SQL_PLAN_BASELINE (
   sql_handle     IN VARCHAR2 := NULL,
   plan_name      IN VARCHAR2 := NULL)
RETURN PLS_INTEGER;


DECLARE
 pls pls_integer ;
BEGIN
 pls := DBMS_SPM.DROP_SQL_PLAN_BASELINE ( 'SYS_SQL_351516f2a705638a') ;
 dbms_output.put_line( pls || '개 삭제' ) ;
END ;
/
```

Baseline 의 삭제는 sql_handle 과 plan_name 을 입력해서 수행한다. 만약 값을 입력하지 않으면 모든 값이 해당된다.

DBA_SQL_PLAN_BASELINES 뷰를 통해 확인

DBA_SQL_PLAN_BASELINES 뷰 조회

```
SQL>SELECT *
SQL>FROM   dba_sql_plan_baselines

SIGNATURE  SQL_HANDLE               SQL_TEXT   PLAN_NAME                      CREATOR
ORIGIN         …
---------- ------------------------ ---------- ------------------------------ -------- -----
3.82498868 SYS_SQL_351516f2a705638a SELECT *   SQL_PLAN_3a58qyamhaswa2b869b05 SYS
MANUAL-LOAD    …
```

DBA_SQL_PLAN_BASELINES 뷰 칼럼 내용

Column	Datatype	Description
SIGNATURE	NUMBER	SQL ID. V$SQL(AREA)뷰의
SQL_HANDLE	VARCHAR2(30)	BASLINE ID
SQL_TEXT	CLOB	SQL text
PLAN_NAME	VARCHAR2(30)	Plan ID
CREATOR	VARCHAR2(30)	BASELINE 생성유저
ORIGIN	VARCHAR2(14)	BASELINE 생성경로
DESCRIPTION	VARCHAR2(500)	BASELINE 설명
VERSION	VARCHAR2(64)	BASELINE 생성시 DB 버젼
CREATED	TIMESTAMP(6)	BASELINE 생성 시간
LAST_MODIFIED	TIMESTAMP(6)	BASELINE 마지막 수정 시간
LAST_EXECUTED	TIMESTAMP(6)	BASELINE 마지막 실행 시간
LAST_VERIFIED	TIMESTAMP(6)	BASELINE 마지막 검증 시간
ENABLED	VARCHAR2(3)	Optimizer 에 의해 사용될 수 있는지를 나타내는 속성
ACCEPTED	VARCHAR2(3)	ACCEPTED 속성
FIXED	VARCHAR2(3)	FIXED 속성
AUTOPURGE	VARCHAR2(3)	사용하지 않고 일정시간이 지날 경우 자동 purge
OPTIMIZER_COST	NUMBER	BASELINE 생성시 COST
MODULE	VARCHAR2(48)	Application 모듈명
ACTION	VARCHAR2(32)	Application Action
EXECUTIONS	NUMBER	BASELINE 생성 시점의 값
ELAPSED_TIME	NUMBER	BASELINE 생성 시점의 값
CPU_TIME	NUMBER	BASELINE 생성 시점의 값
BUFFER_GETS	NUMBER	BASELINE 생성 시점의 값
DISK_READS	NUMBER	BASELINE 생성 시점의 값
DIRECT_WRITES	NUMBER	BASELINE 생성 시점의 값
ROWS_PROCESSED	NUMBER	BASELINE 생성 시점의 값
FETCHES	NUMBER	BASELINE 생성 시점의 값
END_OF_FETCH_COUNT	NUMBER	BASELINE 생성 시점의 값

SQL_PLAN_BASELINES 속성 변경을 통한 사용 실행계획 제어

- SPB 속성 변경 패키지 내용

```
DBMS_SPM.ALTER_SQL_PLAN_BASELINE (
   sql_handle        IN VARCHAR2 := NULL,
   plan_name         IN VARCHAR2 := NULL,
   attribute_name    IN VARCHAR2,
   attribute_value   IN VARCHAR2)
 RETURN PLS_INTEGER;
```

- 변경 가능 속성 : enabled, fixed, autopurge, plan_name, description

Baseline 의 실행계획 사용시 우선순위를 결정하는 속성으로 ENABLED, ACCEPTED, FIXED 의 3 가지 속성이 있다.

위 세가지 속성 중에서 ENABLED 와 ACCEPTED 속성은 Baseline 이 사용 가능한지를 나타내는 속성으로 모두 'YES' 상태인 Baseline 만 사용이 가능하다. ENABLED 속성은 사용자가 직접 변경 가능하다. ACCEPTED 속성의 경우 최초 등록되는 Baseline 과 사용자가 커서 캐시에서 수동으로 등록하는 Baseline 은 'YES'상태로 등록된다. 실행계획 변경이 발생하여 자동 등록되는 실행계획의 경우 'NO' 상태로 등록 된다.

FIXED 속성은 Baseline 사용시 우선 순위를 결정하는 속성으로 ENABLED 와 ACCEPTED 속성이 모두 'YES'인 Baseline 중 FIXED 속성이 'YES'인 Baseline 이 우선적으로 선택된다. 만일 우선순위가 같은 Baseline 이 여러 개 존재할 경우에는 Optimizer 에 의해 Cost 가 낮게 판단되는 Baseline 이 선택된다.

일단 Baseline 에 등록되어 사용 가능한 Baseline 이 존재하고, DB 파라메터가 Baseline 을 사용하도록 설정되어 있으면 SQL 실행시 Baseline 을 통해 실행계획을 반영한다. 이후에 SQL 외적인 요인으로 SQL 의 실행계획에 변화가 생길 경우 새로 생성되는 실행계획은 바로 SQL 수행에 반영되지 않고, ACCEPTED 속성이 'NO'인 상태로 자동으로 SPB 에 저장된다. 새로 생성된 Baseline 은 검증과정을 거쳐 ACCEPTED 속성이 'YES'인 상태로 바뀔 때까지 Baseline 선택에서 제외된다.

■ SPB 검증 패키지

```
DBMS_SPM.EVOLVE_SQL_PLAN_BASELINE (
   sql_handle   IN VARCHAR2 := NULL,
   plan_name    IN VARCHAR2 := NULL,
   time_limit   IN INTEGER  := DBMS_SPM.AUTO_LIMIT,
   verify       IN VARCHAR2 := 'YES',
   commit       IN VARCHAR2 := 'YES')
  RETURN CLOB;
```

패키지를 실행하면 non accepted 상태의 Baseline 에 대해 검증 작업을 수행한다. Verify 값이 yes 일 경우 time_limit 으로 설정된 시간 이내에 실제 검증 작업을 수행한다. Commit 이 yes 일 경우 accepted 속성을 yes 로 바꾸고, no 일 경우 바뀌지 않는다. 결과 값으로 검증 작업에 대한 레포트를 생성한다.

■ 테스트

```
SQL>SELECT signature ,
SQL>       sql_handle ,
SQL>       plan_name ,
SQL>       origin ,
SQL>       enabled ,
SQL>       accepted ,
SQL>       fixed
SQL>FROM   dba_sql_plan_baselines

SIGNATURE SQL_HANDLE                      PLAN_NAME                      ORIGIN
ENABLED ACCEPTED FIXED OPTIMIZER_COST
---------- ---------------------- ---------------------- -------------- ------- ---
3.82498868 SYS_SQL_351516f2a705638a   SQL_PLAN_3a58qyamhaswa2b869b05 AUTO-CAPTURE
YES     NO       NO              1938
3.82498868 SYS_SQL_351516f2a705638a   SQL_PLAN_3a58qyamhaswaeaf1df04 AUTO-CAPTURE
YES     YES      NO                 2
```

현재 아래쪽 Baseline 을 사용중 새로운 실행계획의 생성으로 위쪽 Baseline 이 생성된 상태. (위 FULL SCAN, 아래 INDEX SCAN)

Accepted 가 NO 인 상태이기 때문에 Baseline 이 사용되지 않는다. YES 로 바꾸기 위한 검증작업 실행

```
DECLARE
 pls clob ;
BEGIN
 pls := dbms_spm.evolve_sql_plan_baseline( 'SYS_SQL_351516f2a705638a' ,
'SQL_PLAN_3a58qyamhaswa2b869b05' , DBMS_SPM.AUTO_LIMIT, 'yes' , 'yes' ) ;
 dbms_output.put_line( pls ) ;
END ;
/
-------------------------------------------------------------------------------
                        Evolve SQL Plan Baseline Report
-------------------------------------------------------------------------------

Inputs:
-------
  SQL_HANDLE = SYS_SQL_351516f2a705638a
  PLAN_NAME  = SQL_PLAN_3a58qyamhaswa2b869b05
  TIME_LIMIT = DBMS_SPM.AUTO_LIMIT
  VERIFY     = yes
  COMMIT     = yes

Plan: SQL_PLAN_3a58qyamhaswa2b869b05
------------------------------------
  Plan was verified: Time used .12 seconds.
  Plan failed performance criterion: 95.07 times worse than baseline plan.

                           Baseline Plan      Test Plan       Stats Ratio
                           -------------      ---------       -----------
  Execution Status:              COMPLETE       COMPLETE
  Rows Processed:                       1              1
  Elapsed Time(ms):                  .159         10.617               .01
  CPU Time(ms):                      .111         10.553               .01
  Buffer Gets:                          3           7153                 0
  Physical Read Requests:               0              0
  Physical Write Requests:              0              0
  Physical Read Bytes:                  0              0
  Physical Write Bytes:                 0              0
  Executions:                           1              1
```

```
------------------------------------------------------------------------------
                                Report Summary
------------------------------------------------------------------------------
Number of plans verified: 1
Number of plans accepted: 0
```

검증작업 실행시 Commit 값을 'yes'로 입력했으나. 실제 검증시에 새로 생성된 SQL 이 비효율적이라 판단되서 인증되지 않았다.

새로운 Baseline 사용하려면 현재 사용되는 Baseline 보다 우선순위를 높게 만들어줘야 한다.
Accepted 값을 바꾸기 위해 실제 Verify(검증)하지 않고 강제 변경

```
------------------------------------------------------------------------------
                        Evolve SQL Plan Baseline Report
------------------------------------------------------------------------------
Inputs:
-------
  SQL_HANDLE = SYS_SQL_351516f2a705638a
  PLAN_NAME  = SQL_PLAN_3a58qyamhaswa2b869b05
  TIME_LIMIT = DBMS_SPM.AUTO_LIMIT
  VERIFY     = no
  COMMIT     = yes

Plan: SQL_PLAN_3a58qyamhaswa2b869b05
------------------------------------
  Plan was changed to an accepted plan.

------------------------------------------------------------------------------
                                Report Summary
------------------------------------------------------------------------------
Number of plans verified: 0
Number of plans accepted: 1
```

Fixed 속성 변경

```
DECLARE
 pls pls_integer ;
BEGIN
 pls := DBMS_SPM.ALTER_SQL_PLAN_BASELINE ( 'SYS_SQL_351516f2a705638a' ,
'SQL_PLAN_3a58qyamhaswa2b869b05', 'fixed', 'yes') ;
 dbms_output.put_line( pls || '개 변경' ) ;
END ;
/
```

```
SIGNATURE SQL_HANDLE                    PLAN_NAME                      ORIGIN
ENABLED ACCEPTED FIXED OPTIMIZER_COST
---------- ---------------------- ------------------------ ------------ ------- -----
3.82498868 SYS_SQL_351516f2a705638a       SQL_PLAN_3a58qyamhaswa2b869b05 AUTO-CAPTURE
YES     YES      YES             1938
3.82498868 SYS_SQL_351516f2a705638a       SQL_PLAN_3a58qyamhaswaeaf1df04 AUTO-CAPTURE
YES     YES      NO                 2

SQL_ID        SQL_TEXT                                SQL_PLAN_BASELINE
PLAN_HASH_VALUE
----------- ------------------------------- -------------------------- --------------
93zny6hkjj6s7 SELECT * FROM   spmtest WHERE  lv = 100  SQL_PLAN_3a58qyamhaswa2b869b05
1690735414
```

- Baseline 사용시 Fixed 된 것이 우선순위가 높기 때문에 해당 Baseline 사용.

만약 2 개 모두 Fixed 될 경우

```
DECLARE
 pls pls_integer ;
BEGIN
 pls := DBMS_SPM.ALTER_SQL_PLAN_BASELINE ( 'SYS_SQL_351516f2a705638a' ,
'SQL_PLAN_3a58qyamhaswaeaf1df04', 'fixed', 'yes') ;
 dbms_output.put_line( pls || '개 등록' ) ;
END ;
/

SIGNATURE SQL_HANDLE                    PLAN_NAME                      ORIGIN
ENABLED ACCEPTED FIXED OPTIMIZER_COST
--------- ---------------------- ------------------------ ------------ ------- -----
3.82498868 SYS_SQL_351516f2a705638a       SQL_PLAN_3a58qyamhaswa2b869b05 AUTO-CAPTURE
YES     YES      YES             1938
3.82498868 SYS_SQL_351516f2a705638a       SQL_PLAN_3a58qyamhaswaeaf1df04 AUTO-CAPTURE
YES     YES      YES                2

SQL_ID        SQL_TEXT                                SQL_PLAN_BASELINE
PLAN_HASH_VALUE
----------- ------------------------------- ------------------------------ -----------
93zny6hkjj6s7 SELECT * FROM   spmtest WHERE  lv = 100  SQL_PLAN_3a58qyamhaswaeaf1df04
1577308861
```

- 우선순위가 같은 Baseline 이 존재할 경우 COST 가 낮은쪽의 Baseline 이 선택된다.

일단 SPB 에 어떤 SQL 에 대한 Baseline 이 존재하면 SQL 의 Text 가 동일한 SQL 에 대해서 실행계획의 변화가 생길 때 해당 실행계획에 대한 Baseline 이 자동으로 등록된다. 다만 SPB 의 FIXED 속성이 'YES'인 Baseline 이 존재하고, 해당 Baseline 이 사용되고 있는 경우에는 SQL 의 실행계획에 영향을 미치는 변화가 생겨도 새로운 실행계획을 생성하지 않는다.

이상의 내용을 다음 표와 같이 정리할 수 있다.

적용여부 / 속성	우선 적용	적용가능	적용안됨	적용안됨
ENABLED	YES	YES	ALL	NO
ACCEPTED	YES	YES	NO	ALL
FIXED	YES	NO	ALL	ALL
새 실행계획	생성안함	SPB 등록(비검증)		

결론

DB 를 운영하는데 있어서 통계정보의 변경이나 인덱스의 상태 변화, DB 의 파라메터 변경, DB version 변경 등 SQL 의 실행계획을 변화 시킬 수 있는 요인은 많이 있다. 또 이로 인한 성능 문제를 겪는 경우도 있을 수 있다. 그런데 이런 문제가 생길 수 있다고 해서 해당 작업들을 아예 하지 않을 수는 없는 일이다. 이럴 때 SQL 의 실행계획 변화에 의한 문제를 막기 위해 SPM 사용을 고려 해 볼 수 있다. SPM 은 여러 개의 실행 계획을 저장해 놓고 유동적으로 사용이 가능하다.

SPM 에 대한 내용을 숙지하고, 적절하게 사용할 수 있다면, DB 를 운영하는데 도움이 될 수 있을 것이다.