Introdução ao SPM (SQL Plan Management Baseline)

Olá, neste artigo iremos falar do passo a passo de como utilizar o SPM para otimizar uma query e fixar um plano de execução desejado.

Links úteis

- Use SPM to Manage SQL Execution Plans
- SQL Plan Management in Oracle Database 19c

O primeiro passo é identificar a query que necessita uma melhora de plano de execução.

Para isso podemos utilizar uma das seguintes queries.

Query problemática:

```
SELECT RECID
FROM TABLE1
WHERE USERNAME LIKE :1 ORDER BY MNEMONIC ASC NULLS FIRST, RECID;
```

- A query acima está executando um FULL TABLE SCAN (FTS) e não está levando em consideração o uso do índice.

Através do SQL_ID:

```
SELECT * FROM GV$SQL_PLAN WHERE SQL_ID = '5wyb7qcb01mxb';
```

```
SELECT * FROM GV$OPEN_CURSOR WHERE SQL_ID = '5wyb7qcb01mxb';
```

```
SELECT * FROM DBA_HIST_SQLBIND WHERE SQL_ID = '5wyb7qcb01mxb' ORDER BY
LAST_CAPTURED;
```

Através do texto ou parte do texto:

```
SELECT * FROM GV$OPEN_CURSOR WHERE SQL_TEXT LIKE '%TABLE1%';
```

```
SELECT * FROM GV$SQL WHERE SQL_TEXT LIKE '%TABLE1%';
```

- O importante nas etapas acima é identificar o **SQL_ID** da query e o **PLAN_HASH_VALUE** do plano de execução que apresenta um problema de performance.

Uma vez identificada a query seja através de seu SQL_ID ou através do texto, podemos então carregar o plano de execução problemático em memória:

```
var res number;
exec :res := dbms_spm.load_plans_from_cursor_cache(sql_id =>
'5wyb7qcb01mxb', plan_hash_value => 2008034826);
```

Podemos checar o plano carregado em memória através das queries abaixo:

```
SELECT * FROM DBA_SQL_PLAN_BASELINES ORDER BY CREATED;
```

```
SELECT * FROM DBA_SQL_PLAN_BASELINES WHERE TO_CHAR(CREATED,'DD/MM/YYYY') =
TO_CHAR(SYSDATE,'DD/MM/YYYY') ORDER BY CREATED;
```

- O objetivo é identificar o SQL_HANDLE to plano de execução problemático através das queries acima.
- No meu caso é o seguinte:
    - sql_handle ⇒ 'SQL_677239092ca3b096'

Uma dica extra para poder carregar o plano de execução problemático caso a query não esteja mais em memória, é utilizar o repositório do AWR.

Query para identificar o range de snapshot no AWR onde a query se encontra com o seu plano de execução:

```
SELECT SNAP_ID,SQL_ID,PLAN_HASH_VALUE FROM DBA_HIST_SQLSTAT WHERE SQL_ID =
'5wyb7qcb01mxb';
```

- Aqui o importante é identificar o range de snapshot:
    - **begin_snap** e **end_snap**

Execução da package do SPM para carregar o plano em memória:

```
var res number
exec :res := dbms_spm.load_plans_from_awr( begin_snap=>1,end_snap=>43,
basic_filter=>q'# sql_id='5wyb7qcb01mxb' and plan_hash_value='2008034826' #'
);
```

Podemos verificar o plano de execução através do SQL_HANDLE:

```
SELECT * FROM TABLE( DBMS_XPLAN.DISPLAY_SQL_PLAN_BASELINE( SQL_HANDLE
=>'SQL_677239092ca3b096', FORMAT => 'basic'));
```

Também podemos verificar em conjunto com o SQL_HANDLE + PLAN_NAME:

```
SELECT * FROM
DBMS_XPLAN.DISPLAY_SQL_PLAN_BASELINE('SQL_677239092ca3b096','SQL_PLAN_6fwjt1
4qa7c4q330d7e98');
```

Com essa informação podemos então gerar/criar um novo plano de execução mais performático ou usar um já existente que apresente uma melhor performance e influenciar o plano problemático para que ele então use o melhor plano de acesso.

Abaixo irei forçar o uso do índice através de um **HINT**:

```
SELECT /*+INDEX (A USERNAME_IDX)*/ RECID
FROM TABLE1 A
```

```
WHERE USERNAME LIKE :1 ORDER BY MNEMONIC ASC NULLS FIRST, RECID;
```

- O **HINT** é uma técnica que permite influenciar o otimizador do banco de dados para que ele leve em consideração a sua recomendação.

Ao executar a query usando o HINT, podemos então identificar o novo plano de execução que faz uso do índice:

```
SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);
```

ou

```
SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY_CURSOR);
```

- Iremos utilizar o SQL_ID e o HASH_VALUE do novo plano de execução para influenciar o plano problemáltico:
  - SQL_ID: 8y3pyv1axd3dv
  - HASH_VALUE: 4238396409

Podemos então o carregar/forçar o bom plano de execução sobre o plano problemático:

```
var res number
exec :res := dbms_spm.load_plans_from_cursor_cache(sql_id =>
'8y3pyv1axd3dv', plan_hash_value => 4238396409, sql_handle =>
'SQL_677239092ca3b096', fixed => 'YES');
```

- O ponto importante aqui, é que identificamos o novo plano de execução (SQL_ID + HASH_VALUE) e iremos influenciar o SQL_HANDLE do plano problemático.

Podemos checar o plano carregado em memória através das queries abaixo:

```
SELECT * FROM DBA_SQL_PLAN_BASELINES ORDER BY CREATED;
```

```
SELECT * FROM DBA_SQL_PLAN_BASELINES WHERE TO_CHAR(CREATED,'DD/MM/YYYY') =
TO_CHAR(SYSDATE,'DD/MM/YYYY') ORDER BY CREATED;
```

Uma vez carregado em memória o bom plano de execução e influenciado o seu SQL_HANDLE problemático, podemos excluir o plano problemático:

```
var res number
exec :res :=DBMS_SPM.DROP_SQL_PLAN_BASELINE
('SQL_677239092ca3b096','SQL_PLAN_6fwjt14qa7c4q330d7e98');
```

- As colunas utilizadas para remoção do plano problemático são as seguintes:
  - SQL_HANDLE e PLAN_NAME

- Até a próxima.

Introdução ao SPM (SQL Plan Management Baseline)

— Autor: *Leonardo Lopes* *03/02/2025 07H:05*