

Problemas com Hints ? SQL Profiles

Uma das muitas melhoras no versão 10g foi a introdução de SQL Profiles
SQL Profiles podem prover o otimizador com informações adicionais, mas eles não determinam um plano de execução como "hints" ou as "stored outlines". A recomendação ou não de SQL Profile acontece quando utilizamos SQL Tuning Advisor em uma consulta lenta.

Esta recomendação pode tornar a consulta bem mais rápida.

Uma situação ideal para a utilização desta característica, é quando você tem comandos em programas que usam hints, que já não são tão rápidos e que não podem ser alterados, uma vez, que são programas/aplicações de terceiros.

Você pode então dar uma dica ☺ para que o otimizador não utilize estes hint's com SQL Profiles. Como no exemplo abaixo:

Criei uma versão maior da tabela SALES, que chamei de SALES2 com a coluna channel_id bem irregular em valores

```
SQL> select channel_id,count(*) from sales2 group by channel_id;
```

```
CHANNEL_ID  COUNT(*)
-----
1           1
2        2064200
4        947328
3       4322624
9         16592
```

Criei histogramas para que o otimizador saiba desta irregularidade.

Também criei um index na coluna channel_id que é acessado através do seguinte comando:

```
SQL> select sum(amount_sold) from sales2 where channel_id=1;
```

```
SUM(AMOUNT_SOLD)
-----
1232,16
```

Plano de Execução

Plan hash value: 1648585356

```
-----
| Id | Operation | Name | Rows | Bytes | Cost (%CPU) | Time |
-----
| 0 | SELECT STATEMENT | | 1 | 15 | 4 (0) | 00:00:01 |
| 1 | SORT AGGREGATE | | 1 | 15 | | |
| 2 | TABLE ACCESS BY INDEX ROWID | SALES2 | 1 | 15 | 4 (0) | 00:00:01 |
|* 3 | INDEX RANGE SCAN | SALES_CHANNEL_ID_IDX | 1 | | 3 (0) | 00:00:01 |
-----
```

Predicate Information (identified by operation id):

```
-----
3 - access("CHANNEL_ID"=1)
```

Mas o que acontece quando mudamos o valor de channel_id para 3 ?

```
SQL> select sum(amount_sold) from sales2 where channel_id=3;
```

```
SUM(AMOUNT_SOLD)
-----
          463002085
```

Plano de Execução

```
-----
Plan hash value: 2862189843
```

```
-----
```

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		1	15	8228 (6)	00:01:39
1	SORT AGGREGATE		1	15		
* 2	TABLE ACCESS FULL	SALES2	4322K	61M	8228 (6)	00:01:39

```
-----
```

Predicate Information (identified by operation id):

```
-----
2 - filter("CHANNEL_ID"=3)
```

Como podemos ver o otimizador escolhe fazer um Full Table Scan (FTS). O que é perfeito. Acessar via index seria muito mais lento, conforme podemos ver na comparação abaixo:

```
SQL> set timing on
SQL> set autot off
SQL> select /*+ index (sales2,SALES_CHANNEL_ID_IDX) */
2 sum(amount_sold) from sales2 where channel_id=3;
```

```
SUM(AMOUNT_SOLD)
-----
          463002085
```

Decorrido: 00:00:30.29

```
SQL> select sum(amount_sold) from sales2 where channel_id=3;
```

```
SUM(AMOUNT_SOLD)
-----
          463002085
```

Decorrido: 00:00:06.62

Vamos assumir agora que esta "HINT", já esta codificada em uma aplicação que acessa meu banco de dados. Como poderemos melhorar a performance, sem ter acesso ao código da aplicação?

É ai que entra o SQL Tuning Advisor.

```
SQL> set timing off
SQL> declare
2 tempstring varchar2(300);
3 task_id varchar2(200);
4 begin
5 tempstring := 'select /*+ index (sales2,SALES_CHANNEL_ID_IDX) */ sum(amount_sold)
6 from sales2 where channel_id=3';
7 task_id := dbms_sqltune.create_tuning_task(sql_text => tempstring,
8 task_name=>'SQLPROFILBLA');
9 dbms_sqltune.execute_tuning_task('SQLPROFILBLA');
```

```
9 end;
10 /
```

Procedimento PL/SQL concluído com sucesso.

O nosso amigo SQL Tuning Advisor recomenda um SQL Profile:

```
SQL> set long 500
SQL> select dbms_sqtlune.report_tuning_task('SQLPROFILBLA') from dual;

SQL> /
```

```
DBMS_SQLTUNE.REPORT_TUNING_TASK('SQLPROFILBLA')
-----
GENERAL INFORMATION SECTION
-----
Tuning Task Name          : SQLPROFILBLA
Tuning Task Owner         : SH
Scope                     : COMPREHENSIVE
Time Limit(seconds)      : 1800
Completion Status         : COMPLETED
Started at                 : 08/29/2009 13:09:22
Completed at              : 08/29/2009 13:09:27
Number of SQL Profile Findings : 1

-----
Schema Name: SH
SQL ID      : 7qpx2gbh80rp0
SQL Text    : select /*+ index (sales2,SALES_CHANNEL_ID_IDX) */
              sum(amount_sold) from sales2 where channel_id=3

-----
FINDINGS SECTION (1 finding)
-----

1- SQL Profile Finding (see explain plans section below)
-----
Foi encontrado um plano de execução potencialmente melhor para esta
instrução.

Recommendation (estimated benefit: 87,96%)
-----
- Considere a aceitação do perfil SQL recomendado.
  execute dbms_sqtlune.accept_sql_profile(task_name => 'SQLPROFILBLA',
    replace => TRUE);
```

Decido então aceitar a recomendação

```
SQL> exec dbms_sqtlune.accept_sql_profile(task_name => 'SQLPROFILBLA',replace=> TRUE);
```

Procedimento PL/SQL concluído com sucesso.

Vamos ver agora o que acontece quando executo o comando com "HINT"

```
SQL> set autotrace on explain
SQL> select /*+ index (sales2,SALES_CHANNEL_ID_IDX) */
          2 sum(amount_sold) from sales2 where channel_id=3;
```

```
SUM(AMOUNT_SOLD)
-----
          463002085
```

Plano de Execução

```
-----
Plan hash value: 2862189843
```

```
-----
| Id | Operation | Name | Rows | Bytes | Cost (%CPU) | Time |
```

0	SELECT STATEMENT		1	15	8228	(6)	00:01:39
1	SORT AGGREGATE		1	15			
* 2	TABLE ACCESS FULL	SALES2	4322K	61M	8228	(6)	00:01:39

Predicate Information (identified by operation id):

2 - filter("CHANNEL_ID"=3)

Note

- SQL profile "SYS_SQLPROF_0148d9bbf23d8000" used for this statement

HINT ignorada !

Este artigo está disponível para download em :

<http://aguimaraes.wordpress.com>