Global Hints

Christian Antognini
Trivadis AG
Zürich, Switzerland

Introduction

Oracle8i Release 2 provides a new feature named *global hints*.

This article shows how to use global hints to tune SQL statements containing views. In fact using hints inside a view can cause the following problems:

- If the view is used in many contexts, it is difficult (and sometimes impossible) to have a good execution plan in all situations.
- Hints specified in a mergeable view and in the SQL statement can lead to conflicts, it is then quite difficult to know which hints are used. In fact the rules are not the same for all hints.
- Hints are never pushed into a nonmergeable view.

No information on SQL tuning or on the optimizer is given. The reader should therefore be able to read an execution plan and be familiar (i.e. know how they work) with terms like: hint, mergeable and nonmergeable view. These subjects and many others are also covered in our course *Optimizer Workshop*.

For a complete description refer to *Oracle8i Designing and Tuning for Performance* manual.

Instead of trying to explain what global hints are using an overly long text let's take a look at an example.

Example

A view BIG_EMP_V is created on top of BIG_EMP (1 million rows). The view gives back to the user only the employees for whom she/he is directly in charge off, or all employees for the president (KING).

```
CREATE OR REPLACE VIEW big_emp_v AS
    SELECT emp1.*
    FROM   big_emp mgr1, big_emp emp1
    WHERE  mgr1.empno = emp1.mgr
    AND    mgr1.ename = user
    AND    user <> 'KING'
    UNION ALL
    SELECT emp2.*
    FROM   big_emp emp2
    WHERE  user = 'KING'
```

To speed up the SQL statements the following indexes has been created.

```
Index Name                     Column Name
------------------------------ ------------------------
BIG_EMP_PK                     EMPNO
BIG_EMP_ENAME_I                ENAME
BIG_EMP_MGR_I                  MGR
BIG_EMP_SAL_I                  SAL
```

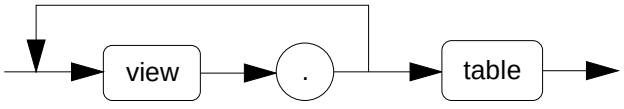
Via an application the team leaders can get a list with the employees in a certain range of salary. The following output shows the SQL statement (:b1 and :b2 are two bind variables) and the corresponding execution plan.

```
SELECT *
FROM big_emp_v
WHERE sal between :b1 and :b2

Execution Plan
----------------------------------------------------------------------
SELECT STATEMENT
  VIEW OF 'BIG_EMP_V'
    UNION-ALL
      FILTER
        HASH JOIN
          TABLE ACCESS(BY INDEX ROWID) OF 'BIG_EMP'
            INDEX(RANGE SCAN) OF 'BIG_EMP_ENAME_I' (NON-UNIQUE)
          TABLE ACCESS(BY INDEX ROWID) OF 'BIG_EMP'
            INDEX(RANGE SCAN) OF 'BIG_EMP_SAL_I' (NON-UNIQUE)
      FILTER
        TABLE ACCESS(BY INDEX ROWID) OF 'BIG_EMP'
          INDEX(RANGE SCAN) OF 'BIG_EMP_SAL_I' (NON-UNIQUE)
```

As you can see the index BIG_EMP_SAL_I is used for both “parts” of the UNION. If the execution plan is not efficient and all statistics are up to date, usually, the only way to solve the problem is to give the optimizer some hints. But in this case “normal” hints are not pushed into the nonmergeable view, thus you have to add the hints directly into the view ☹. To solve such problems the global hints can be used.

To specify a global hint you must use the extended syntax for the table name or alias. Is also possible to specify references for many levels (for view in view). The objects must simply be separated by a dot.



Here an example of the same SQL statement using global hints to “force” another execution plan.

```
SELECT /*+ index(big_emp_v.emp1 big_emp_mgr_i) full(big_emp_v.emp2) */
       *
FROM big_emp_v
WHERE sal between :b1 and :b2

Execution Plan
----------------------------------------------------------------------
SELECT STATEMENT
  VIEW OF 'BIG_EMP_V'
    UNION-ALL
      FILTER
        NESTED LOOPS
          TABLE ACCESS(BY INDEX ROWID) OF 'BIG_EMP'
            INDEX(RANGE SCAN) OF 'BIG_EMP_ENAME_I' (NON-UNIQUE)
          TABLE ACCESS(BY INDEX ROWID) OF 'BIG_EMP'
            INDEX(RANGE SCAN) OF 'BIG_EMP_MGR_I'
      FILTER
        TABLE ACCESS(FULL) OF 'BIG_EMP'
```

As you can see, it is possible to reference the objects inside the view. With these references the optimizer is able to push the hints into the right view.

Conclusion

Global hints are another nice feature that can be used to influence the behavior of the optimizer. Unfortunately, using hints is not a solution but a workaround, because each time you have to write a hint in your SQL statements, the conclusion is:

The optimizer can’t, do it yourself!