Adaptive Query Optimization

DOAG Datenbank, Düsseldorf, 3 June 2014

Christian Antognini
@ChrisAntognini

- Senior principal consultant, trainer and partner at Trivadis in Zurich (CH)
  - christian.antognini@trivadis.com
  - http://antognini.ch

- Focus: get the most out of Oracle Database
  - Logical and physical database design
  - Query optimizer
  - Application performance management


- Proud member of OakTable Network, Oracle ACE Director
Introduction

- Over the years Oracle has extremely improved the capabilities of the query optimizer

- Most of the improvements fall into one of the following areas
  - Enhance the quality of the inputs (e.g. objects statistics)
  - Make the gathering and management of object statistics easier and more efficient
  - Implement or enhance query transformations
  - Improve plan stability
  - Cope with poor estimations that leads to poor execution plans

- 12c is no exception, every one of these areas were improved
Adaptive Query Optimization Is a Set of Features

- Adaptive plans (Enterprise Edition only)
  - Join methods
  - Parallel distribution methods
- Adaptive statistics
  - SQL plan directives
  - Automatic reoptimization
    - Statistics feedback (evolution of *cardinality feedback*)
    - Performance feedback
  - Dynamic statistics (evolution of *dynamic sampling*)
Enables or disables adaptive query optimization features
- Adaptive plans
- SQL plan directives
- Automatic reoptimization
  - It isn’t the case in 12.1.0.1 (bug 16824474)

The default value is TRUE
Enables or disables **reporting mode** for adaptive query optimization features

Useful to assess how an execution plan would change

Use **DBMS_XPLAN** to get detail information about the analysis

- Might fail with an ORA-1001 (bug 17270605)

```sql
SELECT *
FROM table(dbms_xplan.display_cursor(format=>'report'))
```

The default value is **FALSE**
Adaptive Plans – Challenge

- Object statistics don’t always provide sufficient information to find an optimal execution plan
- To get additional insights, the query optimizer can take advantage of features like dynamic sampling and cardinality feedback
  - Don’t solve all issues, though
As of 12c, the query optimizer can **postpone some decisions until the execution phase**

The idea is to leverage information collected while executing part of an execution plan to determine how another part should be carried out

The query optimizer uses adaptive plans in two situations:
- To switch the **join method** from a nested loops join to a hash join
- To switch the **PX distribution** method from hash to broadcast
Adaptive Plans – Concept (2)

- The query optimizer adds *subplans* to execution plans
  - One of the alternatives is the *default plan*

- A subplan is chosen during the *first execution* based on the number of rows actually processed
  - The query optimizer computes an *inflection point*

- A new row source operation is used to partially buffer and count the rows
  - STATISTICS COLLECTOR

- The selection of the *final plan* is only performed during the first execution
Adaptive Plans – Join Method Switch Example

```
SELECT * FROM t1, t2 WHERE t1.id = t2.id AND t1.n = 666
```

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>HASH JOIN</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>NESTED LOOPS</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>NESTED LOOPS</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>STATISTICS COLLECTOR</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>TABLE ACCESS FULL</td>
<td>T1</td>
</tr>
<tr>
<td>6</td>
<td>INDEX UNIQUE SCAN</td>
<td>T2_PK</td>
</tr>
<tr>
<td>7</td>
<td>TABLE ACCESS BY INDEX ROWID</td>
<td>T2</td>
</tr>
<tr>
<td>8</td>
<td>TABLE ACCESS FULL</td>
<td>T2</td>
</tr>
</tbody>
</table>
Adaptive Plans – Join Method Switch Example

```sql
SELECT * FROM t1, t2 WHERE t1.id = t2.id AND t1.n = 666
```

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>HASH JOIN</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>NESTED LOOPS</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>NESTED LOOPS</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>STATISTICS COLLECTOR</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>TABLE ACCESS FULL</td>
<td>T1</td>
</tr>
<tr>
<td>6</td>
<td>INDEX UNIQUE SCAN</td>
<td>T2_PK</td>
</tr>
<tr>
<td>7</td>
<td>TABLE ACCESS BY INDEX ROWID</td>
<td>T2</td>
</tr>
<tr>
<td>8</td>
<td>TABLE ACCESS FULL</td>
<td>T2</td>
</tr>
</tbody>
</table>
Adaptive Plans – Join Method Switch Example

```sql
SELECT * FROM t1, t2 WHERE t1.id = t2.id AND t1.n = 666
```

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>HASH JOIN</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>NESTED LOOPS</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>NESTED LOOPS</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>STATISTICS COLLECTOR</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>TABLE ACCESS FULL</td>
<td>T1</td>
</tr>
<tr>
<td>6</td>
<td>INDEX UNIQUE SCAN</td>
<td>T2_PK</td>
</tr>
<tr>
<td>7</td>
<td>TABLE ACCESS BY INDEX ROWID</td>
<td>T2</td>
</tr>
<tr>
<td>8</td>
<td>TABLE ACCESS FULL</td>
<td>T2</td>
</tr>
</tbody>
</table>
Adaptive Plans – V$SQL.IS_RESOLVED_ADAPTIVE_PLAN

- NULL means that the execution plan associated to the cursor isn’t adaptive
- N means that the final execution plan hasn’t been determined
- Y means that the final execution plan was determined
  - Also set if reporting mode is enabled
Adaptive Plans – Searching for Inflection Point of Join Method

- For both join methods, the cost associated to different cardinalities is estimated
  - The cardinality of the outer table varies
  - The cardinality of the inner table remain fixed
- The query optimizer uses a binary search
- The search takes place between a minimum and maximum cardinality
Adaptive Plans – Searching for Inflection Point Example
SQL Plan Directives

- SQL plan directives are automatically created when misestimates occur
  - They are temporarily stored in the SGA and flushed to disk every 15 min
- They aren’t tied to a specific SQL statement, but to specific columns
  - Several of them can be used for a single SQL statement
- They instruct the database engine to automatically create extended statistics
  - Only column groups are considered
  - If creating extended statistics isn’t supported/possible, they instruct the query optimizer to use dynamic sampling
- The database engine automatically maintains them
  - Some functionalities are exposed through DBMS_SPD
### SQL Plan Directives – Data Dictionary

- **DBA_SQL_PLAN_DIRECTIVES**
- **DBA_SQL_PLAN_DIR_OBJECTS**

```sql
SQL> SELECT type, reason, count(*)
    2   FROM dba_sql_plan_directives
    3   GROUP BY type, reason;

<table>
<thead>
<tr>
<th>TYPE</th>
<th>REASON</th>
<th>COUNT (*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DYNAMIC_SAMPLING</td>
<td>SINGLE TABLE CARDINALITY MISESTIMATE</td>
<td>81</td>
</tr>
<tr>
<td>DYNAMIC_SAMPLING</td>
<td>JOIN CARDINALITY MISESTIMATE</td>
<td>180</td>
</tr>
<tr>
<td>DYNAMIC_SAMPLING</td>
<td>GROUP BY CARDINALITY MISESTIMATE</td>
<td>6</td>
</tr>
</tbody>
</table>
```
Statistics Feedback

- Evolution of **cardinality feedback**
- Used for **single-table** cardinalities as well as **join** cardinalities
- Information about misestimates might be persisted through SQL plan directives
  - For misestimates due to table functions no information is stored
- **V$SQL.IS_REOPTIMIZABLE**
  - Y means that the next execution will trigger a reoptimization
  - N means that no reoptimization is necessary
  - R means that reoptimization information is available, but reporting mode was enabled
Dynamic Statistics

- Evolution of **dynamic sampling**
- Used for **single-table** cardinalities as well as **join** and **group-by** cardinalities
- **OPTIMIZER_DYNAMIC_SAMPLING** has a new level: **11**
  - At this level the query optimizer decides when and how to use dynamic statistics
  - Several new features are enabled only at this level
- Insights resulting from dynamic statistics can be shared and persisted through **SQL plan directives**
Summary

- Adaptive query optimization is one of the most interesting features of 12c
- The query optimizer is getting more and more dynamic
Questions and answers ...

Christian Antognini
Senior Principal Consultant
christian.antognini@trivadis.com