International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-4 Issue-6, August 2015

A Novel Approach for Finding Optimal Query Plan in RDBMS

Shani S. Das, Rejimoan.R

Abstract— Information must be organized in such a way that it is able to access, update and manage. Database is a collection of such information that is organized in a well structured manner. Since databases allow flexible data storage, huge amount of information can be stored in it. Structured Query Language (SQL) is used to do database operations especially the retrieval of data inside database. The database operations must not be too time consuming. Hence the database operations must be done in an efficient and effective manner. The existing optimizer relies on cost as well as heuristic approach. Our focus is to find an optimal execution plan for a query.

Index Terms— SQL Query, Query Optimizer, Optimal Query Plan

I. INTRODUCTION

Database contains large amount of data. The data is retrieved from database basically through SQL statements [1]. The SQL statements are declarative, and hence different procedures can be used to write the SQL queries to get the same result. But the use of optimal query is essential because the performance of any system relies on the data retrieval. Hence SQL Query Tuning [2] is essential. Each database contains a query compiler to execute a query. The query optimizer which is a part of query compiler generates an optimized strategy for query execution which is then used by the query complier to execute the query in an efficient way. Basically the optimizer is either Cost- based or Heuristic-based[3] [4].

The query plan is the sequence of steps that must be performed in a ordered manner to access the data in the database. The query optimizer is not much perfect to always generate optimized query plans for a query. If any database users or the administrators find that, the query submitted is consuming a lot of time then he must be able to load the execution plans manually. SQL Plan Management is used to manually load the plans based on the required efficiency .The main characteristic of the SQL Plan Management is that, eventhough it allows users to manually load the execution plans, it does not allow the user to degrade the existing performance of query execution. In this paper we are manually loading multiple plans for a single query and find the best query plan for that query.

Manuscript published on 30 August 2015.
* Correspondence Author (s)
Shani.S.Das*, Department of Computer Science and Engineering, SCTCE, Trivandrum, Kerala, India.
Rejimoan.R, Department of Computer Science and Engineering, SCTCE, Trivandrum , Kerala, India.

II. RELATED WORKS

The optimizers function is to develop the optimized query plan for the query. Different optimizers exist and these optimizers use basically two types of approach for the execution plan generation- the cost based and heuristic based. The cost based optimizers have several cost components to consider. They include access cost refers to the secondary storage ,Storage cost[6] refers to the cost for storing the intermediate results, Computation cost which include the coast for the CPU and Memory transfers and finally the communication cost. The working of the cost based optimizers is that the cost of each plan is estimated, it is a number that represents the estimated resource usage for an execution plan. Based on the estimation, the plan with the lowest estimated cost is chosen as the optimal one. The Heuristic based approach applies certain set of predefined rules to the query and develop an optimal plan. Sometimes the optimizers fail to load the optimal execution plan. LEO optimizer (LEarning Optimizer) [6] [8] of DB2 works by exploiting this. The optimizer will check each time the plan developed by the optimizer is optimal or not. If the plan is not a good one then the optimizer will correct its statistics and generate a new plan. This plan is compared with the previous plan. If the new plan is accurate the feedback loop exists in the optimizer that will use this new plan for the upcoming sql queries. The feedback loop is used to enhance the available information on the database where the most queries have occurred is chosen. This allows the optimizer to actually learn from its past mistakes. Complex queries are optimized using case base reasoning [7] in which the similarity level is used to compare the query with the existing one.

III. PROPOSED SYSTEM

The proposed method focuses on finding an optimal plan for the query which reduces the execution time of the user submitted query. The optimal plan for a query is find out by manually executing the each plan for the query. Different execution plans are manually generated and each plan is executed individually. The optimized plan is estimated based on time. The minimum time for a execution plan is the criteria for identifying it as the optimized plan. SQL Plan Management SQL Plan Management ensures that runtime performance will never degrade due to the change of an execution plan. In light of this if any inefficient plans are executed or the query retrieval time get too long then there is a possibility to load plans manually to the optimizer. These plans are stored into datastructure. Each plan is individually executed. The execution time of a query using a particular plan is the criteria for selecting the optimal plan for the query. The plan with less execution time is the optimal one.

Inorder to manually generate the query execution plan the first step is to identify the actual execution plan and disable it. The administrator will force the optimizer to generate different plans for the same query by supplying hints. For example, in a join query of three tables Employee, Department and Project, the administrator can force the optimizer to generate the plan based on the index of either Employee table or project table .Similar to the INDEX hint, MERGE is another hint that can be used to generate another plan for the same query. Each hint generates each plan. INDEX hint tells the optimizer to use the index of the particular table to generate the query execution plan. The MERGE hint is used when the referenced table become the inner table for the join operation. That is, forces the optimizer to use sort merge join. Hence by supplying different hints for the same query, the administrator can force the optimizer to generate different plans. This process is repeated for different queries and all these plans are stored in a heap structure. The generated plans for the same query are identified using the heap structure. If the administrator needs to retrieve all generated plans for a particular query, this handle value is used.

As any database system, Oracle is also a database which mainly focuses on storing and retrieval of huge amount of information. Oracle provides [5] different database versions by which each of these versions are upward compatible. Oracle 11g introduced a new feature of SQL Plan Management which helps to manually load different plans for the query. The plan generation module is tested in this SQL Plan Base provided by Oracle 11g.This provides different methods to load the query plans - From Staging Table, From Cursor cache, From Stored outlines and Using SQL Tuning Set. Out of these methods Cursor cache is used for loading the plans. The following steps describe how different plans are developed by using hints.

- Execute the SQL query so that a plan is generated by the optimizer
- Retrieve the SQL_ID of the query
- Load the plan from the cache
- Retrieve the SQLHANDLE name and plan name from the heap
- Disable the plan by using the plan name
- Give a hint to the optimizer for the same query
- Retrieve the SQL_ID and plan hash value
- Execute the plan by supplying SQLPLAN_ID and SQL HANDLE value from cursor cache
- Repeat the process by giving new hints to the optimizer so that multiple plans can be loaded to baseline

For example,

Consider a schema which contains Employee, Department and Project Tables. Suppose a user need to retrieve the project number, department by which the project is developing, and the employee name, address and birthdate who is doing the project in the corresponding department. Then its SQL query is,

1. SELECT P.Pnumber, P.Dnum,E.Lname, E.Address, E.Bdate FROM PROJECT AS P, DEPARTMENT AS D, EMPLOYEE AS E WHERE P.Dnum=D.Dnumber and D.Dnumber=E.Dno;
2. The required SQL_ID is retrieved using the query

SELECT sql_id, sql_fulltext FROM V$SQL WHERE sql_text LIKE '%SELECT PROJECT.Pnumber%';

3. After retrieving the SQL_ID use the SQL_ID to load the available plans from the cursor cache .For example the SQL_ID is 7tdzmaxw181qm

EXECUTE: cnt: =DBMS_SPM.LOAD_PLANS_FROM_CURSOR_CACHE (sql_id ='7tdzmaxw181qm');

4. To load the plans it is needed to know the SQLHANDLE because the SQLHANDLE uniquely identifies different plans for a single query.

SELECT sql_handle, sql_text, plan_name, enabled
FROM dba_sql_plan_baselines;

Let SQLHANDLE value is SYS_SQL_265b531f5c95ac32

5. Give a hint to the optimizer to generate another plan. For example the hint is INDEX then the query becomes

SELECT /*+INDEX (EMPLOYEE)*/ P.Pnumber, P.Dnum, E.Lname, E.Address,E.Bdate FROM PROJECT AS P, DEPARTMENT AS D, EMPLOYEE AS E WHERE P.Dnum=D.Dnumber and D.Dnumber=E.Dno;

6. Repeat the step 2 to get the SQL_ID of the new plan generated by the INDEX hint. Let the SQL_ID of this plan is a31zxkc5cm8a2
7. Load plans for the cusror cache using the SQL_ID in the above step and the handle name from the step 4

Exec:cnt:=dbms_spm.load_plans_from_cursor_cache (sql_id = 'a31zxkc5cm8a2', plan_hash_value = 2774714633, sql_handle = ('SYS_SQL_265b531f5c95ac32') ;

8. Repeat the steps 5, 6, and 7 with another hint to generate another plan for the same query
9. The loaded plans are viewed from the baselines using

select t.*from table (dbms_xplan.display_ sql_ plan_baseline('SYS_SQL_265b531f5c95ac32', format = 'basic')) t;

10. The execution time of a query can be found by set timing on or using the query

SELECT CPU_TIME, ELAPSED_TIME FROM V$SQLAREA WHERE SQL_FULLTEXT LIKE '%SELECT /*+INDEX (EMPLOYEE) %';

IV. Results and Analysis

As discussed above, the optimal plan is obtained by comparing different query execution plans for that query. One among those query plans for comparison is the plan initially developed by the optimizer, when the query is submitted. The optimal plan is chosen by executing the query based on the plan and obtaining its execution time. The query plan with lowest execution time is taken as the optimal query plan. Some of the queries and its optimal plan are shown below.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

SQL QUERIES	HINTS with ELAPSED TIME (msec) OPTIMIZED PLAN								
L-LEADING,H-HASH,I-INDEX,O-ORDERED,P-PARALLEL,A-ALLROWS,F-FIRSTROWS,OP-OPTIMIZER,U-USE MERGE									
	L	O	I	A	F	U	H	P	OP	
SELECT P.Pnumber, P.Dnum, E.Lname, E.Address,E.Bdate FROM PROJECT AS P, DEPARTMENT AS D, EMPLOYEE AS E WHERE P.Dnum=D.Dnumber and D.Dnumber=E.Dno;	0.042		).51	-	-	0.6	-	0.45	0.05	ORDERED (O)
SELECT SUM (Salary), MAX (Salary), MIN (Salary), AVG (Salary) FROM EMPLOYEE;	-		0.02	0.01	-	-	0.012	-	0.013	ALL_ROWS(A)
SELECT ssn, Fname, Address FROM EMPLOYEE, DEPARTMENT WHERE Dnumber=Dno;	-		0.012	0.012	-	-	-	0.09	0.01	OPTIMIZER(OP)
SELECT E.Fname, E.Lname FROM EMPLOYEE AS E WHERE EXISTS (SELECT * FROM DEPENDENT AS D WHERE E.Ssn=D.Essn AND E.Sex=D.Sex AND E.Fname = D.Dependent_name);	0.013		0.012	-	-	0.01	-	0.05	0.04	USE_MERGE(U)
SELECT Fname, Lname FROM EMPLOYEE WHERE EXISTS (SELECT * FROM DEPENDENT WHERE Ssn=Essn) AND EXISTS (SELECT * FROM DEPARTMENT WHERE Ssn=Mgr_ssn);			0.035	-	-	0.043	-	0.03	0.08	0.04	HASH(H)

V. CONCLUSION

The performance of any application which uses databases indirectly depends on the data retrieval from that database. Every data retrieving queries execute based on the execution strategy developed by the query complier. Sometimes the query optimizer may fail to produce an optimized query plan and hence the finding an optimized query plan is so important. The work mainly focuses on the reducing the time for the query execution. The query execution is more important database operation which affects the working of the critical applications. If the database operation related to the application is time consuming then it becomes a cumbersome. To reduce this, our work relies on reducing the query execution time by generating more optimized plans.

REFERENCES

1. P. Prof.M.A.Pund, S.R.Jadhao, "A role of query optimization in relational database," International Journal of Scientific & Engineering Research, Volume 2, Issue 1,pp - 1-7, January2011.
2. Dr. G. R. Bamnote, "Introduction to query processing and optimization," International Journal of Advanced Research in Computer Science and Software Engineering , Volume 3, Issue 7, pp : 6-13, July 2013.
3. Y.Ioannidis, "Query optimization," in Proc ACM Computer Survey,vol. 28, pp. 121–123, 1998.
4. S. Chaudhuri, "An overview of cost-based optimization of queries with aggregates," IEEE DE Bulletin pp : 34-43, Sep1995.
5. An oracle white paper november 2010 "sql plan management in oracle database 11g" 2013.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

6. Volker Markl Mokhtar Kandil Michael Stillger, Guy Lohman. Leo – db2's learning optimizer. Proceedings of the 27th VLDB Conference, Roma, Italy,, 2001.
7. Sakshi Mathur Pragya Shukla. Optimizing complex queries using casebased reasoning with dynamicity management. IEEE Proceedings, 2014.
8. V. Raman V. Markl, G. M. Lohman. Leo: An autonomic query optimizer for db2. IBM SYSTEMS JOURNAL, VOL 42, NO 1,, 2003.
9. C. A. van den Berg and M. L. Kersten. "Analysis of a dynamic query optimization technique for multijoin queries" Elsevier Science Inc., 2010.

Shani .S.Das is currently doing MTech in Computer Science and Engineering in Sree Chitra Thirunal College of Engineering under Kerala University, Trivandrum , Kerala, India. Shani received BTech Degree in Computer Science and Engineering from Govt. Engg. College under M.G University , Kottayam. Kerala, India in 2011. She concentrates mainly on database, machine learing. She has also worked as Guest Lecturer in Information Technology in Govt. Engg. College , Painavu , Idukki Kerala, India.

Rejimoan. R is working as professor at department of computer science and engineering, Sree Chitra Thirunal College of Engineering, Trivandrum, Kerala. Now he is doing his research in Machine Leraning. He has published his research works in many national and international conferences and journals. He has working experience of 11 years in Sree Chitra Thirunal College of Engineering. He is interested in Databases, Machine learning and Natural Language Processing.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)