IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 10, Issue 4 (Mar. - Apr. 2013), PP 38-40
www.iosrjournals.org

A Study on Optimization of Top-k Queries in Relational Databases

Neha Singh, P.K. Pandey*
Department of Computer Science, A.P.S. University, Rewa (M.P.)
** Department of Physics, Govt. Science College, Rewa (M.P.)*

Abstract: *Efficient execution of top-k queries is increasingly becoming a major challenge for relational database technology. Top-k query has emerged as a key requirement in modern application. In these applications, efficient and adaptive evaluation of top-k queries is an integral part of application semantics. In this paper we discuss a query optimization framework of top-k queries that fully integrates rank-join operators. Ranking queries produce results that are ordered on some computed score. A key property of top-k queries is that, users are interested only in the first k results and not in total ranking of all query results. This property directly impacts the optimization of top-k queries by optimizing for the first k results. Traditionally, most real world database systems offer the feature of first n row optimization.*

I. Introduction

The increasing importance of top-k queries warrants an efficient support of ranking in the relational database management system. Emerging applications that depend on ranking queries warrant efficient support of ranking in database management system. Integration of database and information retrieval technologies ahs been an active research topic recently [1]. Top-k queries are dominant in many emerging applications as web databases, multimedia databases, middlewares, datamining etc. Supporting ranking gives database systems the ability to efficiently answer Information Retrieval queries. A top-k query is an important type of query that allows for supporting IR applications on top of database systems. Database system provide guarantee of efficient data handling with solid integrity and consistency. An important subject in this connection is determining the suitable size of inputs in relation N for answering to the rank ware query so that in this manner top-k query answers are gotten. It is vital in information integration with large scale to select top-k rank aware from multiple sources and it has also basic role for minimizing transfer cost because as the size of relations become smaller, transfer cost become less [6]. The algorithm that is used for answering to the ranking queries is estimating input sizes by means of statistical relations, monotony hypothesis and random variables [2, 3]. The other new innovation provides a systematic and principled framework to support efficient evaluations of top-k queries in relational database systems by extending relational algebra and query optimization [4]. In this connection one another solution is improving the join and using the ripple join thath minimize the time in order to get estimation with relatively acceptable precision for query results. The main idea of ripple join is join algorithm aware the suggested rank for supporting join queries with top-k relational database [5]. In this paper we study the optimization of top-k queries.

II. Relational Database

A Relational database management system (RDBMS) is a database management system (DBMS) that is based on the relational model as introduced by E. F. Codd. A relational database of data items organized as a set of formally described relations (also known as table) from which data can be accessed or reassembled in many different ways without having to reorganize the database relations.

III. Top-K Queries

The top-k query is define as- Given a database D of m objects, each of which is characterized by n attributes, a scoring function f, according to which we rank the objects in D, and the number of expected results k. Then a top-k query Q returns the k objects with the highest rank in f. In Top-k query, query define on n attribute a1, a2 ,…, an and relation M in the form of R1 ,R2 ,…,RM that each ai (i=1:n) belongs to one relation Rj (j=1:M). Each of the attributes has special domain in comparison with their kind. According to the query, a series of attributes of these relations are applied for projection, a series of attributes of these relations are used for restriction and join. In the rank aware queries there is apart for ranking that some of relations attribute are presented in the form of a ranking relation which is called ranking function. Ranking function f is formed in the form of attribute n' that is n' <=n. A theory for ranking function f is this: ranking function changes in comparison with all relations are monotonic. In addition to this, the number of suitable answers in rank aware queries is determined too that is just Top k. Consider a set of relations R1 to Rm. Each tuple in Ri is associated with some score that gives it a rank within Ri. The top-k join query joins R1 to Rm and produces the results ranked on a total score. The total score is computed according to some function, say F that combines individual scores. Note that the score attached with each relation can be the value of one attribute or a value computed using a predicate on a subset of its attributes. A possible SQL-like notation for expressing a top-k join query is as follows:

```
SELECT *
FROM R1, R2, . . . ,Rm
WHERE join condition (R1, R2, . . . ,Rm)
ORDER BY F (R1.score, R2.score. . . Rm.score)
LIMIT k;
```

Where LIMIT limits the number of results reported to the user.

IV. Query Optimization

Query optimization is necessary for high level relational queries to provide an opportunity for the database system to systematically evaluate alternative query execution strategies and to choose an optimal strategy. Query optimization is a function of many relational database management systems in which multiple query plans for satisfying a query are examined and a good query plan is identified. The function of query optimization is to select an efficient execution strategy. The figure 1 shows the query optimization process which generates the best query execution plan. For a given query, the search space can be defined as the set of equivalent operator tress that can be produced using transformation rules.

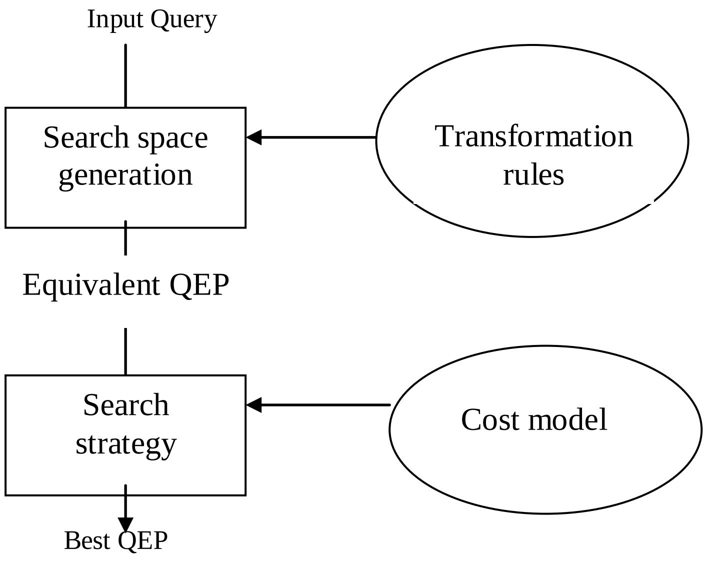

figure 1: Query Optimization Process

V. Optimization of Top-k Queries

Optimization of top-k queries (also known as Rank aware query optimization) in relational database has contributed to the existing system for the query operations significantly. In this section we discuss a rank aware query optimization framework that fully integrates rank-join operators into relational query engine and how to extend the traditional query optimization to handle the new rank join operators. The two major task are includes while integrating the new rank-join operators in query optimizer- enlarging the space of possible plans to include those plans that use rank join operators as a possible join alternative and providing a costing mechanism for the new operators to help the optimizer prune expensive plans in favor of more general cheaper plans.

The first task, enlarging the plan space is achieved by extending the enumeration algorithm to produce new execution plans. Now we exhibit how to extend the enumeration space to generate top-k query execution plans. Top-k plans will integrate the rank-join operators into general execution plans. For a query with n input relations, R1 to Rn, assume there exists a ranking function f(s1,s2,…sn), where si is score expression on relation Ri. For two sets of input relations, P and Q, we extend the space of plans that join P and Q to include rank join plans by make suitable for following:

- Join Eligibility: P and Q are rank-join eligible if all the following apply:

1. There is a join condition that relates at least one input relation in P to an input relation in Q.

2. f can be expressed as f(f1(SP); f2(SQ); f3(SO)), where f1, f2 and f3 are three scoring functions, SP are the score expressions on the relations in P, SQ are the score expressions on the relations in Q, and SO are the score expressions on the rest of the input relations.
3. There is at least one plan that accesses P and/or Q ordered on SP and/or SQ, respectively.
-Join Choices: Rank-join can have several implementations as physical join operators. For each rank-join between P and Q, plans can be generated for each join implementation.
-Join Order: For symmetric rank-join operators there is no distinction between outer and inner relations. For the nested-loops implementation, a different plan can be generated by switching the inner and the outer relations.

Now we discuss the second task, providing a costing mechanism for the new operators to help the optimizer prune expensive plans in favor of more general cheaper plans. In real-world query optimizers, the cost model for different query operators is quite complex and depends on many parameters. Parameters include input cardinality, available buffers, type of access paths etc. Although cost models can be very complex, a key ingredient of accurate estimation is the accuracy of estimating the size of intermediate results. In traditional join operators, the input cardinalities are independent of the operator itself and only depend on the input subplan. Mostly, the output cardinality depends only on the size of the inputs and the selectivity of the logical operation. On the other hand, since a rank-join operator does not consume all of its inputs, the actual input size depends on the operator itself and how the operator decides that it has seen enough information from the inputs to generate the top k results. Hence, the input cardinality depends on the number of ranked join results requested from that operator. Therefore, the cost of a rank-join operator depends on the following:
- The number of required results k and how k is propagated in the pipeline
- The number of tuples from inputs that contain enough information for the operator to report the required number of answers, k.
- The selectivity of the join operation affects the number of tuples propagated from the inputs to higher operators through the join operation. Hence, the join selectivity affects the number of input tuples required by the rank-join operator to produce ranked results.

There are two ways to produce plans that join two sets of input relations, P and Q, and produce ranked results- by using rank-join operators to join L and R subplans or by gluing a sort operator on the cheapest join plan that joins P and Q without preserving the required order. The challenge is in comparing two plans when one or both of them are rank-join plans. For all rank-join plans, the cost of the plan depends on k and the join selectivity s. Since these two parameters are the same for all plans, the pruning among these plans follows the same mechanism as in traditional cost based pruning. For example, pruning a rank-join plan in favor of another rank-join plan depends on the input cardinality of the relations, the cost of the join method, the access paths, and the statistics available on the input scores. We assume the availability of an estimate of the join selectivity, which is the same for both sort-plans and rank-join plans.

VI. Conclusion

Query optimization is essential for large information retrieval system. In this paper we describe the top-k queries and general query optimization method for to provide an opportunity for the database system to systematically evaluate alternative query execution strategies and to choose an optimal strategy. We also discussed optimization of top-k queries with a framework for integrating rank-join operators in real-world query optimizers. Our framework was based on two main steps- first, extend the enumeration phase of the query optimizer to generate rank-aware plans. The extension was achieved by providing rank-join operators as possible join choices and by defining ranking expressions as a new physical plan property.

References

[1] S. Chaudhury, R. Ramakrishnan and G. Weikum (2005), Integrating db and ir technologies, CIDR.
[2] Ilyas, I.F. and W.G. Aref (2006), Adaptive Rank aware Query Optimization in Relational Databases, ACM Transactions on Database Systems.
[3] Ilyas, I.F. and R. Shah (2004), Rank-aware Query Optimization,SIGMOD, June13-18, Paris, France.
[4] Ilyas, I.F. and C. Li, 2005. Ranksql: Query algebra and optimization for relational top-k queries.SIGMOD 2005, June 1416, Baltimore, Maryland, USA.
[5] Ilyas, I.F. and W.G. Aref, 2003. Supporting Top-k Join Queries in Relational Databases. Proceedings of the 29th VLDB Conference, Berlin, Germany.
[6] M.A. Kashem, A S Chowdhury, R Deb, M Jahan (2010), Query Optimization on Relational Databases for Supporting Top-k Query Processing techniques, JCIT