INTRODUCTION TO QUERY PROCESSING AND OPTIMIZATION IN DISTRIBUTED DATABASE SYSTEM

Anjali Saini

1 Department of Computer Science and Engineering, Ph.D Scholar, Singhania University, India
angel.anjali43@gmail.com

ABSTRACT:

Query processing is a technique for achieving information from the database in an effective and convenient way. The performance of the database system rely on the techniques of query processing that we used in various database systems. The query optimizer is a remarkable component or element in today’s relational database management system (RDBMS). This element is creditable for translating or converting a user-submitted query which is generally written in a non-procedural language into an reliable query evaluation program that can be implemented against the database. This research paper describe and give an overview of query processing and optimization steps in distributed database system. The study in this research paper will give the guidelines to the scholars, researchers and practitioners of computer science and engineering in their field.

Keywords- Query Processing, Distributed Database System and Query Optimization

[1] INTRODUCTION

Query processing and optimization is a basic and fundamental part of any Distributed database Management system (DBMS). To be utilized fully and effectively, the queries results must be available in the timeframe required by the submitting user— it may be a person, another separate and distinct DBMS or even robotic assembly machine. In this paper, how a DBMS processes the queries and the methods or techniques it uses to optimize the performance are the topics that will be covered. Distributed data processing is becoming and emerging as a reality now. Businesses want to do it for various reasons, and they often do it in favor of staying...
INTRODUCTION TO QUERY PROCESSING AND OPTIMIZATION IN DISTRIBUTED DATABASE SYSTEM

competitive. At the same time, much of the infrastructure for distributed data processing is already a base e.g., modern network technology, a number of question or problems make distributed data processing still a complex and difficult task. There are three phases that a query passes through during the Database Management System’ processing[1]:

1. Parsing and translation
2. Optimization
3. Evaluation

Most of the queries that are submitted to a DBMS are in a high-level language (HLL) such as SQL. During the parsing and translation stage, the human or user readable form of the query is translated and converted into forms usable by DBMS. These can be in the pattern of a relational algebra expression, query graph and query tree. After parsing and translation into a relational algebra expression, the query is then transformed into a form, usually a query graph or tree, that can be managed by the optimization engine. The optimization engine then do various analyses on query data, that results in generating a number of valid or accurate evaluation plans. From there, it determines the most appropriate or we can say accurate evaluation plan to execute. After the evaluation plan has been selected, it is now passed into the DMBS’ query-execution engine [1] (also called as the runtime database processor [2]), where the plan is executed or completed and the results are returned.

Distributed System composed of multiple independent computers that communicate or share through a computer network (CN). These computers interact or contact with each other in order to carry out a common goal. The computer application or program which runs on a distributed system is known as distributed program. The process or method of writing the distributed program is known as Distributed Programming. It is a kind of system where the information processing is divided over many computers rather than restricted to single machine. Distributed systems are the internet (net) based applications which have two types of computer systems-Client node and Server node. Optimization of query is a function of relational database management system (RDBMS). Generally, the query optimizer cannot be used directly by users, once queries are given to database server, and then parsed by the parser, after that they are passed to the query optimizer where the optimization take place. Queries results are generated by accessing relevant data from the database and then manipulate it in a way that gives the requested information[3]. Since database (DB) structures are complicated, in most cases, and especially for not a very simple queries, the data which are needed for a query that can be collected from a database by accessing it in many different ways through different data-structures, and are in different orders. It determines or tell the lowest cost plan for executing query. By "lowest cost plan," we mean an access path to the data which takes the least amount of time[4].

[2] RELATED WORK

Many researchers worked on Query processing in distributed system from last many years in various languages. Some of the work of earlier authors are listed below-

Researchers and practitioners have been interested in distributed database systems since the 1970s. At that time, the main focus was on supporting distributed data management for large corporations and organizations that kept their data at different offices or subsidiaries. Although
there was a clear need and many good ideas and prototypes (e.g., System [Williams et al. 1981], SDD-1 [Bernstein et al. 1981], and Distributed Ingres [Stonebraker 1985]), the early efforts in building distributed database systems were never commercially successful [Stonebraker 1994]. Transaction processing has been thoroughly investigated in, for example, Gray and Reuter [1993].

Work on data models (relational, deductive, object-oriented, and semi-structured) is described in Ullman [1988], Cattell et al. [1997], Abiteboul [1997], and Buneman [1997]. Also, we will assume that the reader is familiar with basic database system concepts, SQL, and the relational data model. Good introductory textbooks are Silberschatz et al. [1997] and Rama krishnan [1997]. Furthermore, there have been several proposals to manage Web sites and query a network of Web pages; see Florescu et al. [1998] for a survey.

In addition, several proposals to manage and query XML data exist (e.g., McHugh and Widom [1999], Abiteboul et al. [1999], and Florescu et al. [1999]). There have been several surveys on distributed query processing; for example, a paper by Yu and Chang [1984] and parts of the books by Ceri and Pelagatti [1984], O’zsu and Valduriez [1999], and Yu and Meng [1997] are devoted to distributed query processing. These surveys, however, are mostly focused on the presentation of the techniques used in the early prototypes of the 1970 and 1980.

[3] QUERY PROCESSING AND OPTIMIZATION

All database systems must be capable for responding to requests from the user for information i.e. processing queries. Obtaining the desired and valuable information from a database system in a predictable and reliable manner is the scientific way of Query Processing. Obtaining these results back in a timely fashion gives out with the Query Optimization technique.

The three phases where a query passes through DBMS processing is given below:
1. Parsing and translation
2. Optimization
3. Evaluation and Execution

3.1 Parsing and translation:
Parsing and translation phase is used for translating or converting the query into internal form. After translating the queries into internal form it is then converted into relational algebra. It is used for syntax checking and also for verification of relations. The job of this phase is to take out the raw tokens from the strings of characters and then translate them to the data elements internally.

The basic job of the parser is to extract the tokens from the raw string of characters and translate them into the internal data elements (i.e. relational algebra operations and operands) and structures (i.e. query graph, query tree) correspondingly. The last job of the parser is to verify the validity of string and syntax of the original query string.

3.2 Optimization:
Here for planning the query, we used to generate a “lowest cost evaluation plan”.

Anjali Saini
3.3 Evaluation and Execution:
Here for performing the query, the query execution engine performs an evaluation plan, executes that plan and also returns its answer to the query. We have multiple ways for query execution. In a sequential fashion for processing a query, the individual operations can be executed in parallel mode as independent processors or threads or as pipelines.

QUERY OPTIMIZATION

The main function of DBMS’ query optimization engine is to search/find an evaluation plan which reduces the overall query’s cost evaluation. We are familiar with the fact that the costs for performing particular operations like join and select can be changed dramatically. Query optimizer is used to give the most accurate and efficient results after query execution using query plans.

Query Optimization Principles:

If we want to know the principles of optimization we should basically identify the “basic building block of system” which go through all the required algorithms. On this basis we have three points independently. They are:
1. QEP Generation
2. Search strategy
3. Cost Function

1. QEP Generation: It trace about the transformations, target language and the source languages. It also defines how to generate the target language from initial query from given source
language. The target language generally reflect the features of run time where QEP is generated or we can say created.

2. Search Strategy: In this, user submitted query be evaluated by various different QEP’s which are used to create various choices to search a good candidate. So, here we define how to find the different set of possibilities by QEP’s. This type is called Search Strategy.

3. Cost Function: For comparing different QEP’s and choosing the best that gives the best, effective and accurate results this phase is used. This module defines the arithmetic formulas which are used to find the cost of execution plans. There is a formula for every different join method, for every different index type access, and in general we can say for every distinct kind of step that can be found in an execution plan, which gives its cost. On given complexity of many of following steps, most of these formulas are quite a simple approximations of what the system actually perform that are based on certain assumptions based on issues such as buffer management, sequential vs. random I/O, disk-cpu overlap etc.

[4] QUERY ALGORITHMS

Finally queries are reduced to a number of file scan operations on the physical file structures. There exist a different access paths, for each and every relational operation where the particular records are required. The Query execution engine for each query have many specialized algorithms which are made to process a particular relational operation and combinations of many access paths. There are many algorithms which are used:

A. Selection Algorithms:

The Select operation must find the data files for related records that meets the selection criteria. Some examples of selection algorithms are given below:

1. Linear search: In this search each and every record from the database files are read and compared with the selection criteria. For this type of searching, the execution cost for non-key attribute is bk, where bk contains number of blocks in that file which represents the given relations r. The average cost of key-attribute is bk/2, having a worst case of bk[5].

2. Binary search: A binary search equality is done on a primary key attribute. It has a worst-case cost of \[\log(bk)\]. This is most efficient and better than linear search as it is used for large number of data records.

3. Search using a primary index on equality:
In this an equality comparison with B+-index on a key attribute will have a worst-case cost of height of the given tree. It also includes retrieving and fetching the record from the data file. An comparison on a non-key attribute must be same leaving the condition of meeting multiple records, in that case, adding the number of blocks that contains the records to that cost is required.

4. Primary index on comparison using search:
When the comparison operators (>, <) are used to retrieve or getting various multiple records from a database file that are sorted by search attribute then the first record that fulfill the condition is found and the total blocks before (<, \[\leq\]) or after (>, \[\geq\]) is locating that first record and its cost is added to it.

B. Join query Algorithms:
INTRODUCTION TO QUERY PROCESSING AND OPTIMIZATION IN DISTRIBUTED DATABASE

The join query algorithm can be executed in a many ways. In case of disk access, the join operations can be very costly and expensive, for implementing and utilizing accurate and efficient join query algorithms it is very important to minimize the query’s execution time.

There are four types of join query algorithms which are given below:

1. **Nested-Loop Join:** It have a inner ‘for loop’ which is nested within an ‘outer for loop’.
2. **Index Nested-Loop Join:** This algorithm is same as the’ Nested-Loop Join’ excluding an index file that is on the inner relation’s join attribute which is used with a database-file scan on each and every index lookup in the inner loop. For utilizing one of the selection algorithms it is essentially an equality selection. Let cs be the cost for the lookup, then the worst-case cost for joining relation r and s is: bk + nr * cs.
3. **Sort-Merge Join:** This algorithm is used to carry out natural joins and equi-joins which is required that each relation (r and s) to be sorted by the common attributes between each relation (r s). However, each record in r and s is scanned only once, so producing a worst and best-case cost of br + bs.
4. **Hash Join:** This hash join algorithm is used to perform natural joins and equi-joins. This algorithm utilizes two hash tabled file structures simultaneously i.e one for each relation that are partition to each relation’s records into sets that containing identical hash values placed on the join attributes. Each relation is scanned and then its corresponding hash table which are built on the join attribute values.

C. Indexes Role

In this, we use the execution time of many operations such as select and join that are reduced with the help of indexes role. Now, let us focus on different types of a structures of index file and their roles in reducing the execution time and overhead:

1. **Dense Index:** Here the Data-files are ordered by the search key and each search key have a separate index record. This structures needs one request to find out the set of contiguous records with desirable search values among the primary prevalence itself.
2. **Sparse Index:** Here Data-file is ordered by the index search key and some of the search key values have corresponding index records. Here with the search key value each and every index record’s data-file pointer points to the first data-file record. While this structure is less efficient and accurate than a dense index to search the desired records, it requires less storage space and less overhead in operations like insertion and deletion.[5]
3. **Primary Index:** In it the data files are ordered by the attribute which is also a search key in the index file. Primary indices can be dense or sparse. This type of index is also known as Index-Sequential File.
4. **Secondary Index:** Here the data file is ordered by attributes that are different from the search key in the index file. In it, Secondary indices must be dense.
5. **Clustering Index:** A two-level index structure which contains the records in the first level having clustering field value in one field and a second field pointing to a block i.e of 2nd level records in the second level. The records in the second level is having only one field which points to an actual data file record or to another 2nd level block.

[5] CONCLUSION

In this paper, we presented a research perspective on the Query System. The most important and relevant functional requirements of a database system is to process queries in an effective and
timely manner. This is very true in case of a very large and critical applications for example in weather forecasting applications, banking systems and aeronautical applications where there are millions and trillions of records that have data and it becomes hard to store and retrieve data from them.

Some of the basic techniques, principles and algorithms of query processing and optimization are mentioned in this paper. One of the most critical functional requirements of a DBMS is its ability to process queries in a timely manner. Thus, a huge deal of research and resources is spent on creating highly efficient, accurate and smart query optimization engines or designs.

REFERENCES


Anjali Saini

58