Retrieval of Data from Very Large Databases Using Apriori Algorithm

1 P MOHAN GANESH, *Assistant Professor*
2 THUMMULURU KAVITHA *Assistant Professor*
3 RVS RATNA KUMAR *Assistant Professor*

[1,2,3]*Dept of IT, Vignan's Institute of Engineering for women's, AP.*

ABSTRACT

Now-a-days most of the Very Large Databases like Medical Databases, Multimedia Database are consuming Petabytes of memory. For extracting required data from this enormous amount of memory so many retrieval techniques were already developed. But all these techniques are time consuming. We can also use indexing supported by all the commercial databases. When we create an index internally it uses B-Tree data structure which is used for efficient retrieval of data. But, Oracle10g is supporting R-Tree indexes also which are more efficient than B-Tree indexes. On Very Large Databases it is not sufficient to create a single index because these Databases consist of large number of fields. Moreover, it is not necessary to create index on each and every field and it is also not possible because it requires large memory area. In very large databases also most of the times the user concentrates on few fields only. So, in our proposed system, the solution is to create some suggested indexes. And as and when the user's query pattern changes the suitable index from the suggested index set is dynamically loaded into the main memory from the local storage. Our proposed technique continuously monitors the query patterns and it creates the suggested indexes using Apriori algorithm and this suggested index set is updated periodically based on the query pattern changes.

KEYWORDS: Very Large Databases, Potential index, Query Pattern, Histogram R-Tree Indexing**.**

1. INTRODUCTION

Now-a-days most databases applications like medical databases and multimedia databases deal with large volumes of data. It is containing so many attributes. As the number of attributes are increasing it is necessary to access the data very efficiently to use it properly. Most of the commercial databases now-a-days are supporting large amounts of data having large number of attributes. These databases are also giving us the facility of creation of indexes to prune out significant portion of the data set that is irrelevant to specific queries. Multidimensional indexing, dimensionality reduction and some index selection tools all could be applied to the problem. However each of these solutions has some problem. The performance of the multidimensional index structures is subject to the curse of dimensionality and rapidly decreases as the number of dimensions increases. The index selection tools [1] of commercial database systems are also having a problem. That is they provide only static indexes. They are targeted toward lower dimensional databases and do not produce results that are optimized for single high dimensional tables.

In most of the high-dimensional database applications, only a small subset of the overall data dimensions [2] is popular for a majority of queries and that recurring patterns of dimensions queried occur. So, we address the high-dimensional database indexing problem by selecting the lower dimensional indexes based on the the query patterns and data. In our approach we are going to consider both data and query patterns and we are developing a new set of low-dimensional indexes to address a large number of expected queries and pruning of data space to answer the queries effectively. But when the current query patterns are substantially different from the query patterns used to recommend the database indexes, the performance of the system will be drastically reduced. So, in our approach as and when the query pattern changes we are calculating the new index set a head of time. So there won't be any delay in accessing the required data from the large hi-dimensional databases.

2. MATHEMATICAL APPROACH

In this section we define the problem of index selection [3] for a multidimensional space by using a query workload. A query workload W consists of a set of queries that select objects within a specified subspace in the domain. Finding the answers to a query when no index is present reduces to scanning all the points in the data set and testing whether the query conditions re met. In this scenario, we can define the cost of answering the query as the time that it takes to scan the data set, that is, the time to retrieve the data pages from the disk. The assumption is that the time spent on performing the I/O dominates the time required to perform the simple bound comparisons. In the case that an index is present, the cost of answering the query can be lower. The index can identify a smaller set of potential matching objects, and only those data in this section, we define the problem of index selection for a multidimensional space by using a query workload. A query workload W consists of a set of queries that select objects within a specified subspace in the data domain. Finding the answers to a query when no index is present reduces to scanning all the points in the data set and testing whether the query conditions are met. In this scenario, we can define the cost of answering the query as

the time that it takes to scan the data set, that is, the time to retrieve the data pages from the disk. The assumption pages containing these objects need to be retrieved form the disk. The degree to which an index prunes the potential answer set for a query determines its effectiveness for the query.

Our problem can be defined as finding a set of indexes I, given a multidimensional data set DS, a query workload W, an optional indexing constraint C and an optional analysis time constraint ta, that provides the best estimated cost over W. In the context of this problem, an index is considered to be the set of attributes that can be used to simultaneously prune the subspace with respect to each attribute. Therefore the attribute order has no impact on the amount of pruning possible. The goal of this work is to develop a flexible index selection framework that can be tuned to achieve effective static index selection and online index selection for high-dimensional data.

For the online index selection, the goal is to develop a system that can recommend an evolving set of indexes for incoming queries over time such that the benefit of index set changes outweighs the cost of making those changes. Therefore, an online index selection system that differentiates between low cost index set changes and higher cost index set changes and can also make decisions about index set changes based on different cost-benefit thresholds is desirable.

In order to measure the benefit of using a potential index over a set of queries, it is necessary to estimate the cost of executing queries with and without the index. To estimate the query cost we conservatively estimate the number of matches associated with using a given index by using a multidimensional histogram abstract representation of the data set. The histogram captures data correlations between only those attributes that could be represented in a selected index.

The cost associated with an index is calculated based on the number of estimated matches derived from the histogram and dimensionality of the index. Increasing the size of the multidimensional histogram enhances the accuracy of the estimated at the cost of an abstract representation size. While maintaining the original query information for later use to determine the estimated query cost, we apply one abstraction to the query workload to convert each query into the set of attributes referenced in the query. We perform frequent item set mining over this abstraction and only consider those sets of attributes that meet a certain support to be potential indexes. By varying the support, we affect the speed of index selection and the ratio of queries that are covered by potential indexes. We further prune the analysis space using association rule mining by eliminating those subsets above a certain confidence threshold. Lowering the confidence threshold improves the analysis time by eliminating some lower dimensional indexes from consideration but can result in recommending indexes that cover a strict superset of the queried attributes. Our technique differs from existing tools in the method that we use to determine the potential set of indexes to evaluate and in the quantization-based technique that we use to estimate query costs. All of the commercial index wizards work in design time. The DBA has to decide when to run this wizard and over which workload. The assumption is that the workload is going to remain static over time, and in case it changes, the DBA would collect the new workload and run the wizard again. The flexibility afforded by the abstract representation that we use allows it to be used for infrequent index selection considering a broader analysis space or frequent online index selection.

2.1 Proposed Solution

Proposed solution for dynamic index selection is to minimize the cost of the queries in the workload, given certain constraints.

2.1.1 Initialize the abstract Representations

Potential index set P. This is a collection of attribute sets that could be beneficial as an index for the queries in the input query workload. This set is computed using traditional data mining techniques. Considering the attributes involved in each query workload to be a single transaction, P Consists of the sets of attributes that occur together in a query at a ratio greater than the input support. Formally, the support of a set of attributes A is defined as

$$S_A = \frac{\sum_{i=1}^{n} \begin{cases} 1 & \text{if } A \le Q_i \\ 0 & \text{otherwise} \end{cases}}{n}$$

where Qi is the set of attributes in the *i*th query and n is the number of queries.

The confidence of and association rule {set of attributes A}→{set of attributes B}, where A and B are disjoint, is defined as

$$C_{A \rightarrow B} = \frac{\sum_{i=1}^{n} \begin{cases} 1 & \text{if } (A \cup B) \le Q_i \\ 0 & \text{otherwise} \end{cases}}{\sum_{i=1}^{n} \begin{cases} 1 & \text{if } A \le Q_i \\ 0 & \text{otherwise} \end{cases}}$$

where Qi is the set of attributes in the *i*th query, and n is the number of queries. To mine the frequent patterns we are using the Apriori algorithm.

Query Set Q**.** This is the abstract representation of the query workload. It is initialized by associating the potential tial indexes that could be beneficial for each query with that query.

Multidimensional Histogram H. An Abstract representation of the data set is created in order to estimate the query cost associated with using each query's possible indexes to answer that query. This representation is in the form of a multidimensional histogram H.

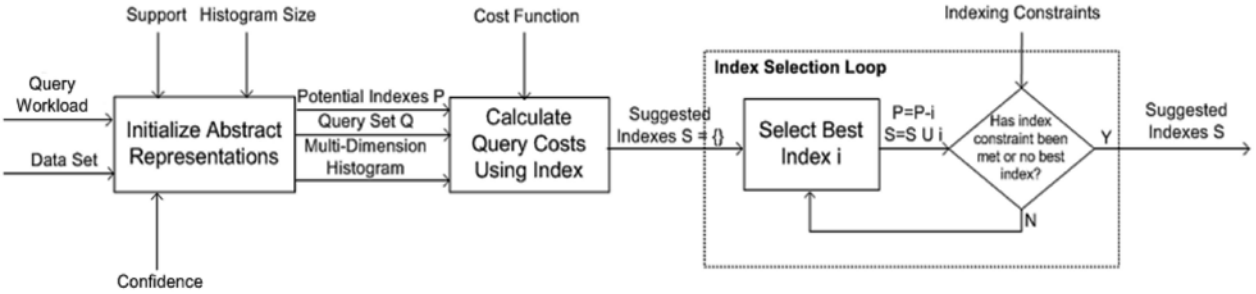

Fig1: Index Selection Flowchart.

The table below shows a simple multidimensional histogram example. This histogram covers three attributes an uses 1 bit to quantize attributes 2 and 3 and 2 bits to quantize attribute 1, assuming that it is queried more frequently than the other attributes. In this example, fro attributes 2 and 3 values form 1 to 5 quantize to 0, and values form 6 to 10 quantize to 1. For attribute 1, values 1 and 2 quantize to 00, 3 and 4 quantize to 01, 5,6, and 7 quantize to 10, and 8 and 9 quantize to 11. The .'s in the column "Value" denote attribute boundaries.

Sample Dataset			
A_1	A_2	A_3	Encoding
2	5	5	0000
4	8	3	0110
1	4	3	0000
6	7	1	1010
3	2	2	0100
2	2	6	0001
5	6	5	1010
8	1	4	1100
3	8	7	0111
9	3	8	1101

Histogram	
Value	Count
00.0.0	2
00.0.1	1
01.0.0	1
01.1.0	1
01.1.1	1
10.1.0	2
11.0.0	1
11.0.1	1

Fig2: Histogram

2.1.2 Query Cost Calculation

Once generated, the abstract representations of the query set Q and the multidimensional histogram H are used to estimate the cost of answering each query by using all possible indexes for the query.For a given query index pair, we aggregate the number of matches that we find in the multidimensional histogram by looking only at the attributes in the query that also occur in the index. To estimate the query cost , we then apply a cost function based on the number of matches that we obtain by using the index and the dimensionality of the index. At the end of this step, our abstract query set representation has estimated costs for each index that could improve the query cost. For each query in the query set representation, we also keep a current cost field, which we initialize to the cost of performing the query by using sequential scan. At this point, we also initialize an empty set of suggested indexes S.

Cost Function. We apply a cost estimate that is base d on the actual matches that occur over the multidimensional histogram over the attributes that form a potential index.

The cost model for R-trees that we use in this work is given by ($d^{(d/2)} * m$), where d is the dimensionality of the index, and m is the number of matches returned for query matching attributes in the multidimensional histogram.

2.1.3 Index Selection Loop

After initializing the index selection data structures and updating estimated query costs for each potentially useful index for a query , we use a greedy algorithm that takes into account the indexes that would be appropriated for the given query work load and data set. For each index in the potential index set P, we traverse the queries in the query set Q that could be improved by that index and accumulate the improvement associated with using that index for that query. After each *i* is selected a check is made to determine if the index selection loop should continue. The input indexing constraints provides one of the loop stop criteria. The indexing constraint could be any constraint such as the number of indexes, total index size, or the total number of dimensions indexed. If no potential index yields further improvement or the indexing constraints have been met, then the loop exits. The set of suggested indexes S contains the results of the index selection algorithm.

2.2 Proposed Solution for Online Index Selection

The online index selection is motivated by the fact that query patterns can change over time. By monitoring the query workload and detecting when there is a change on the query pattern that generated the existing set of indexes, we are able to maintain good performance as query patterns evolve.

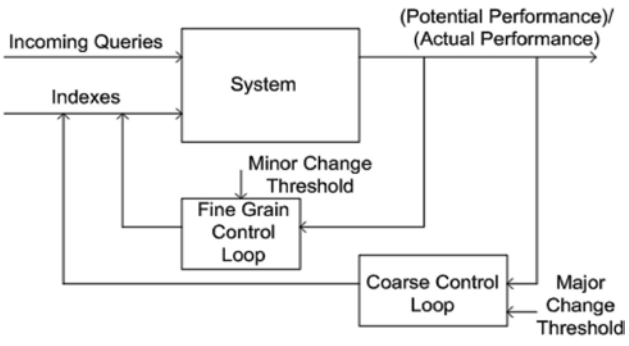

Fig3: Dynamic Index Analysis Framework.

We use control feed back to monitor the performance of the current set of indexes for incoming queries and determine when adjustments should be made to the index set. In a typical control feedback system, the output of a system is monitored, and based on some functions involving the input and output, the input to the system is readjusted through a control feedback loop.

Our system input is set of indexes and a set of incoming queries. Our system simulates and estimates costs for the execution of incoming queries. System output is the ratio of the potential system performance to the actual system performance in terms of database page accesses to answer the most recent queries. We implement two control feedback loops. One is for fine-grained control and is used to recommend minor inexpensive changes to the index set. The other loop is for coarse control and is used to avoid very poor system performance by recommending major index set changes. Each control feedback loop has decision logic associated with it.

3. RESULTS AND DISCUSSION

3.1 Experimental Setup

Data Sets. Several data sets were used during the performance of experiments. The variation in the data sets is intended to show the applicability of our algorithm to a wide range of data sets and to measure the effect that data correlation has on results. The data sets used include the following.

1. Random. This is a set of 1,00,000 records consisting of 100 dimensions of uniformly distributed integers between 0 and 999. The data is not correlated.
2. Stocks. This is a set of 6,500 records consisting of 360 dimensions of daily stock market prices. This data is extremely correlated.
3. Mlb. This is a set of 33,619 records of major league pitching statistics between the years 1900 and 2004 and consists of 29 dimensions of data. Some dimensions are correlated with each other, whereas others are not all correlated.

Analysis Parameters. The effect of varying several analysis input parameters, including support, multidimensional histogram size, and online indexing control feedback decision thresholds, war analyzed. Unless otherwise specified, the confidence parameter for the experiments is 1.0.

Query Workloads. The following query workloads are used in our experiments.

1. Synthetic. This includes 500 randomly generated queries. The distribution of the queries over the first 200 queries is 20 percent involving attributes {1,2,3,4} together, 20 percent {5,5,7}, 20 percent, {8,9}, and the remaining queries involve between one and five attributes that could be any attribute. Over the last 300 queries, the distribution shifts to 20 percent covering attributes{11,12,13,14}, 20 percent {15,16,`7}, 20 percent {18,19}, and the remaining 40 percent are between one and five attributes that could be any attribute.
2. Clinical. This include 659 queries executed from a cli8nical application. The query distribution file has 64 distinct attributes.
3. Hr. This includes 35,860 queries executed from a human resources application. The query distribution file has 54 attributes. Due to the size of this query set, some initial portion of the queries are used for some experiments.

Results are as shown below.

Data Set/ Workload	Tool	Analysis Time(s)	% Queries Improved	Number of Indexes
stock/ clinical	AutoAdmin	450	100	23
	Proposed	110	100	18
stock/ hr	AutoAdmin	338	100	20
	Proposed	160	100	16
mlb/ clinical	AutoAdmin	15	0	0
	Proposed	522	87	16

Fig: Comparison of the Proposed Index Selection Algorithm with AutoAdmin in Terms of Analysis Time and Percentage of Queries improved.

Support	Query/Index Pairs			Total Improvement		
Confidence	100	50	0	100	50	0
2	229	229	90	57710	57710	57603
4	198	198	85	55018	55018	55001
6	185	185	73	46924	46924	46907
8	178	178	66	42533	42533	42516
10	170	170	58	37527	37527	37510

Fig: Comparision of Analysis Complexity and Query Performance as Support and Confidence Vary for the Stock Data Set Using the Clinical Workload.

Fig: Comparative cost of online indexing as the multidimensional histogram size changes for the random data set using the synthetic query workload.

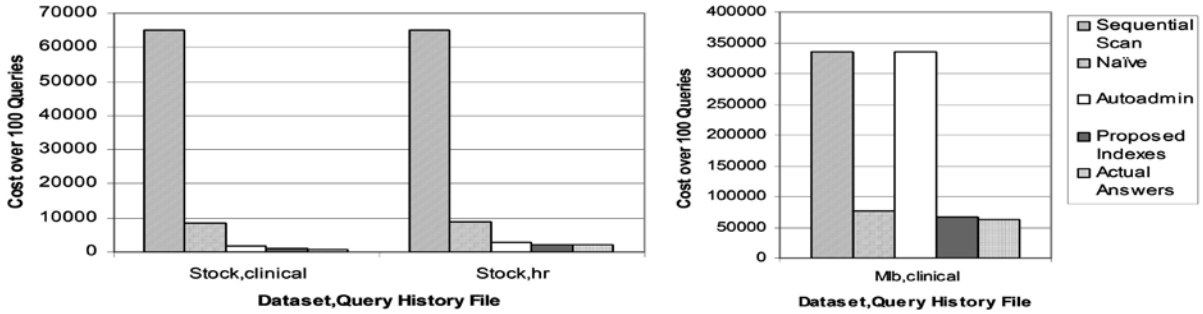

Fig: Costs in data object accesses for ideal, sequential scan, AutoAdmin, naïve and the proposed index selection techniques using relaxed constraints.

4. CONCLUSIONS

A flexible technique for index selection is introduced, which can be tuned to achieve different levels of constraints and analysis complexity.

REFERENCES

[1] A. Dogac, A.Y.Erisik, and A.Ikinici, "An Automated Index Selection Tool for Oracle&: Maestro 7," Technical Report LBNL/PUB-3161, Software Research and Development Center, Scientific and Technical Research Council of Turkey(TUBITAK), 1994.

[2] S.Ponce, P.M. Vila, and R.Hersch, " Indexing and Selection of Data Items in Huge Data Sets by Constructing and Accessing Tag Collections," Poc.19th IEEE Symp. Mass Storage Systems and 10th Goddard Conf. Mass Storage Systems and Technologies, 2002.

[3] S.Choenni, H. Blanken, and T. Chang, "On the Selection of Secondary Indexes in Relational Databases," Data and Knowledge Eng., 1993.