# Online Index Recommendations: A Case Study

[1]Dr. L Prasannakumar, [2]M Sree Jagadeesh, [3]D Ratna Giri, [4]B Radha Krishna
[1,2]Dept. of CSE, DIET College, Visakhapatnam, AP, India
[3]Dept. of IT, SRKR Engineering College, Bhimavaram, West Godavari, AP, India
[4]WOLLO University, Ethiopia

**Abstract**
Large databases pose a challenge with respect to efficient access. Users are usually interested in querying data over a relatively small subset of the entire attribute set at a time. A potential solution is to use lower dimensional indexes that accurately represent the user access patterns. So we are going to design one tool to address these issues we introduce a parameterizable technique to recommend indexes based on index types that are frequently used for large data sets. If the users query pattern changed the index will automatically adjust it. To do that we incorporate a query pattern change detection mechanism to determine when the access patterns have changed enough to warrant change in the physical database design. We perform experiments with a number of data sets, query sets, and parameters to show the effect that varying these characteristics has on analysis results.

**Keywords**
Large Databases, Query Processing, Data Mining

## I. Introduction

An increasing number of database applications such as business data warehouses and scientific data repositories deal with high-dimensional data sets. As the number of dimensions/attributes and the overall size of data sets increase, it becomes essential to efficiently retrieve specific queried data from the database in order to effectively utilize the database. Indexing support is needed to effectively prune out significant portions of the data set that are not relevant for the queries. Multidimensional indexing, dimensionality reduction, and Relational Database Management System (RDBMS) index selection tools all could be applied to the problem. However, for high-dimensional data sets, each of these potential solutions has inherent problems. To illustrate these problems, consider a uniformly distributed data set of 1,000,000 data objects with several hundred attributes. Range queries are consistently executed over five of the attributes. The query selectivity over each attribute is 0.1, so the overall query selectivity is 1=105 (that is, the answer set contains about 10 results). An ideal solution would allow us to read from the disk only those pages that contain matching answers to the query. We could build a multidimensional index over the data set so that we can directly answer any query by only using the index[1-3]. However, the performance of multidimensional index structures is subject to Bellman's curse of dimensionality and rapidly degrades as the number of dimensions increases. For the given example, such an index would perform much worse than a sequential scan. Another possibility would be to build an index over each single dimension. The effectiveness of this approach is limited to the amount of search space that can be pruned by a single dimension.

### A. Terminology

#### 1. High Dimensional Indexing

A number of techniques have been introduced to address the high-dimensional indexing problem such as the X-tree [4] and the GC-tree [5]. Although these index structures have been shown to increase the range of effective dimensionality, they still suffer performance degradation at higher index dimensionality.

#### 2. Feature Selection

Feature selection techniques are a subset of dimensionality reduction targeted at finding a set of untransformed attributes that best represent the overall data set. These techniques are also focused on maximizing data energy or classification accuracy rather than query response[7]. As a result, selected features may have no overlap with queried attributes.

#### 3. Index Selection

The index selection problem has been identified as a variation of the Knapsack Problem, and several papers proposed designs for index recommendations based on optimization rules. These earlier designs could not take advantage of modern database systems' query optimizer [8]. Currently, almost every commercial RDBMS provides the users with an index recommendation tool based on a query workload and uses the query optimizer to obtain cost estimates. A query workload is a set of SQL data manipulation statements. The query workload should be a good representative of the types of queries that an application supports.

#### 4. Automatic Index Selection

The ideas of having a database that can tune itself by automatically creating new indexes as the queries arrive have been proposed. In a cost model is used to identify beneficial indexes and decide when to create or drop an index at runtime. Costa and Lifschitz propose agent-based database architecture to deal with an automatic index creation. Microsoft Research has proposed a physical-design alerter to identify when a modification to the physical design could result in improved performance.

## II. Background

Many excellent studies on data mining have been conducted, such as those reported in [9-13]. [9] considers the problem of inferring classification functions from samples; [10] studies the problem of mining association rules between sets of data items; [12] proposes an attribute oriented approach to knowledge discovery; [9] develops a visual feedback querying system to support data mining; and [3] includes many interesting studies on various issues in knowledge discovery such as finding functional dependencies between attributes. However, most of these studies are concerned with knowledge discovery on non-spatial data, and the study most relevant to our focus here is [13] which studies spatial data mining.
More specifically, [14] proposes a spatial data dominant knowledge extraction algorithm and a non spatial data-dominant one, both of which aim to extract high-level relationships between spatial and non spatial data. However, both algorithms suffer from the following problems. First, the user or an expert must provide the algorithms with spatial concept hierarchies, which may not be available in many applications. Second, both algorithms conduct their spatial exploration primarily by merging regions at a certain level of the hierarchy to a larger region at a higher level. Thus,

the quality of the results produced by both algorithms relies quite crucially on the appropriateness of the hierarchy to the given data. The problem for most applications is that it is very difficult to know a priori which hierarchy will be the most appropriate.
Discovering this hierarchy may itself be one of the reasons to apply spatial data mining. To deal with these problems, we explore whether cluster analysis techniques are applicable. Cluster Analysis is a branch of statistics that in the past three decades has been intensely studied and successfully applied to many applications. To the spatial data mining task at hand, the attractiveness of cluster analysis is its ability to find structures or clusters directly from the given data, without relying on any hierarchies. However, cluster analysis has been applied rather unsuccessfully in the past to general data mining and machine learning. The complaints are that cluster analysis algorithms are ineffective and inefficient. Indeed, for cluster analysis algorithms to work effectively, there need to be a natural notion of similarities among the “objects” to be clustered. And traditional cluster analysis algorithms are not designed for large data sets, say more than 2000 objects.

## III. Our Contribution

### A. Index Selection

Index Selection is a method of artificial selection in which several useful traits are selected simultaneously. First, each trait that is going to be selected is assigned a weight, the importance of the trait. I.e., if you were selecting for both height and the darkness of the coat in dogs, if height was more important to you, one would assign that a higher weighting. For instance, heights weighting could be ten and coat darkness’ could be two.

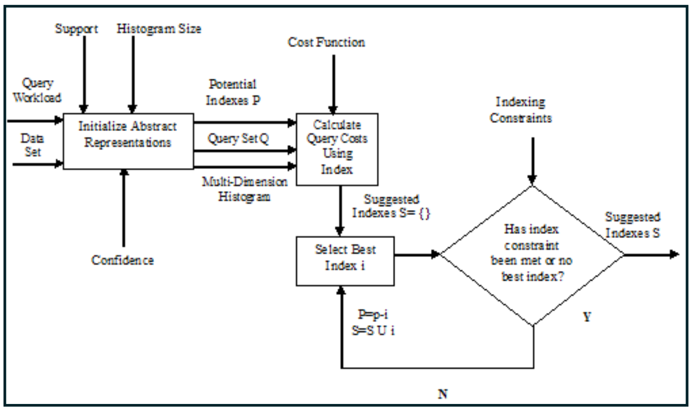


Fig. 1: Index Selection Procedure

This weighting value is then multiplied by the observed value in each individual animal and then the score for each of the characteristics is summed for each individual. This result is the index score and can be used to compare the worth of each organism being selected. Therefore, only those with the highest index score are selected for breeding via artificial selection. This method has advantages over other methods of artificial selection, such as tandem selection, in that you can select for traits simultaneously rather than sequentially. Thereby, no useful traits are being excluded from selection at any one time and so none will start to reverse while you concentrate on improving another property of the organism. However, its major disadvantage is that the weightings assigned to each characteristic are inherently quite hard to calculate precisely and so require some elements of trial and error before they become optimal to the breeder.

### B. Query Access Pattern

The advantage of using data access objects is the relatively simple and rigorous separation between two important parts of an application which can and should know almost nothing of each other, and which can be expected to evolve frequently and independently. Changing business logic can rely on the same DAO interface, while changes to persistence logic do not affect DAO clients as long as the interface remains correctly implemented. In the specific context of the Java programming language, Data Access Objects can be used to insulate an application from the particularly numerous, complex and varied Java persistence technologies, which could be JDBC, JDO, EJB CMP, Hibernate, or many others. Using Data Access Objects means the underlying technology can be upgraded or swapped without changing other parts of the application.

## IV. Implmentation

This paper divides the implementation into following modules

### A. Module 1: Initialize the Abstract Representation

In this module we are monitoring the user queries and initialize the abstract representation. In this module we collection the user transaction and from that we are finding the frequently selected item. And by applying the association rule we are calculation the relationship between the records and finding the support and confidence. Based on that we are initializing the abstract representation. The initialization step uses a query workload and the data set to produce a set of Potential Indexes P, a Query Set Q, according to the support, confidence, and histogram size specified by the user.
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 fro the input 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 where Qi is the set of attributes in the ith query, and n is the number of queries. For instance, if the input support is 10 percent and attributes 1 and 2 are queried together in greater than 10 percent of the queries, then a representation of the set of attributes {1, 2} will be included as a potential index. As the input support is decreased, the number of potential indexes increases. Such an index will only be more effective in pruning data space for those queries that involve only the subset’s attributes. In order to enhance analysis speed with limited effect on accuracy , the input confidence is used to prune the analysis space.
Confidence is the ratio of a set’s occurrence to the occurrence of a subset. While data mining the frequent attribute sets in the query workload in determining P, we also maintain the association rules for disjoint subsets and compute the confidence of these association rules. The confidence of an association rule is defined as the ratio that the antecedent (left-hand side of the rule) and consequent (right-hand side of the rule) appear together in a query, given that the antecedent appears in the query. Although the Apriori algorithm was appropriate for the relatively low attribute query sets in our domain, a more efficient algorithm such as the FP-Tree could be applied if the attribute sets associated with queries are too large for the Apriori technique to be efficient.

### B. Module 2: Calculate the Query Cost

The query cost will be calculated based on the potential index and Query Set. The query will be used to find the best index. We say the index is best one if it give result for all the querys.. 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. The cost estimate provided is conservative in that it will provide a result that is at least as great as the actual number of matches in the database. By evaluating the number of matches over the set of attributes that match the query, the multidimensional subspace pruning that can be achieved using different index possibilities is taken into account.

### C. Module 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 were already selected to iteratively select indexes that would be appropriate for the given query workload and data set. For each index in the potential index set P, we traverse the queries in query set Q that could be improved by that index and accumulate the improvement associated with using that index for that query. The improvement for a given query-index pair is the difference between the cost for using the index and the query's current cost. If the index does not provide any positive benefit for the query, no improvement is accumulated. The potential index i that yields the highest improvement over the query set Q is considered to be the best index. The recommendation set can be pruned in order to avoid recommending an index that is no useful in the context of the complete solution.

### D. Module 4. Calculate the Performance

For each response of the query we are calculating the Performance. Based on that performance the index modification will be performed.

### E. System Input

The system input is made up of new incoming queries and the current set of indexes I, which is initialized to be the suggested indexes S from the output of the initial index selection algorithm. For clarity, a notation list for the online index selection.

The system simulates query execution over a number of incoming queries, that is, the abstract representation of the last w queries stored as W, where w is an adjustable window size parameter. W is used to estimate the performance of a hypothetical set of indexes Inew against the current index set I. This representation is similar to the one kept for query set Q in the static index selection. In this case, when a new query q arrives, we determine which of the current indexes in I most efficiently answers this query and replace the oldest query in W with the abstract representation of q. We also incrementally compute the attribute sets that meet the input support and confidence over the last w queries. This information is used in the control-feedback-loop decision logic. The system also keeps track of the current potential indexes P and the current multidimensional histogram H.

### F. System Output

In order to monitor the performance of the system, we compare the query performance using the current set of indexes I to the performance using a hypothetical set of indexes Inew. The query performance using I is the summation of the costs of queries using the best index from I for the given query.

## V. Objectives

Project objective is to lessen the progressive performance failure and by applying the association rule we reduce the size of the index that accurately represent the user access patterns. Present system having following properties.

- Query response does not perform well if query patterns change.
- Because it uses static query workload.
- Its performance may degrade if the database size gets increased.
- Tradition feature selection technique may offer less or no data pruning capability given query attributes.

We develop a flexible index selection frame work to achieve static index selection and dynamic index selection for high dimensional data.

- Through this a database could benefit from an index change.
- The index selection minimizes the cost of the queries in the work load.
- Online index selection is designed in the motivation if the query pattern changes over time.
- By monitoring the query workload and detecting when there is a change on the query pattern, able to evolve good performance as query patterns evolve.

Finding the answers to the query, when no index is present, reduces to scanning all the points in the dataset and testing whether the query conditions are met takes more amount of time to retrieve the data.

## VI. Testcase

### A. Test Case-1 ( It is shown in fig. 2)

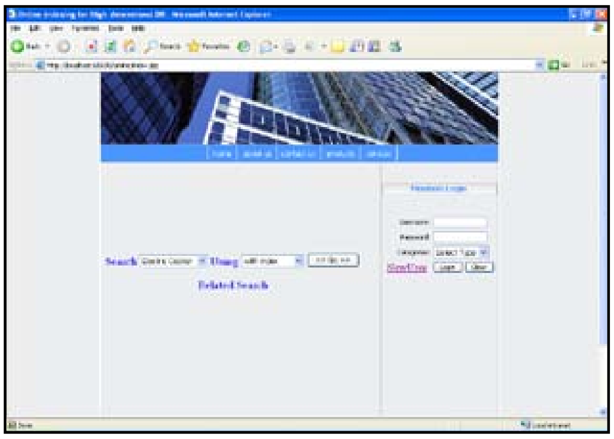
Objective: Index Selection

#### 1. Input Specification

- If Client types correct search type and index information
- If Client gives empty index and searchtype

#### 2. Output Specification

- Clients get data from the database
- Clients get Popup with alert message.



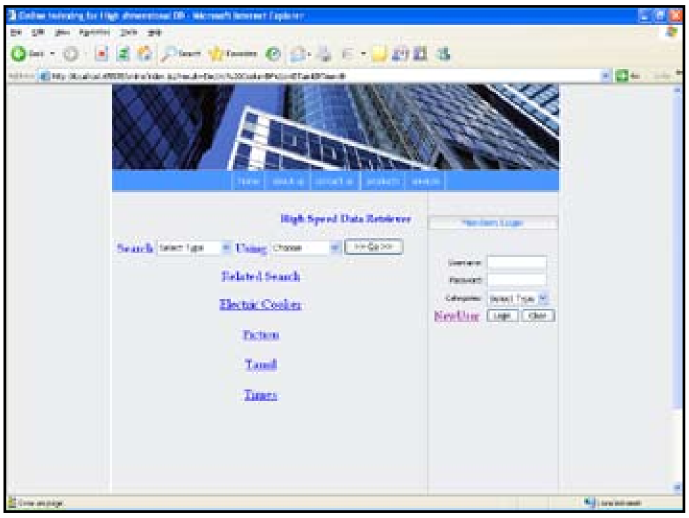


Fig. 2: Related Search Pages

## VII. Result Analysis

After performing various results on various data sets we study the results shown in Table 1,2,3 and fig 3,4,5,6

Table 1: Support and Confidence Calculation

| ITEMS | SUPPORT | CONFIDENCE |
|---|---|---|
| Electric Cooker & Fiction | 26.0% | 66.0% |
| Electric Cooker & Tamil | 33.0% | 83.0% |
| Electric Cooker & Timex | 26.0% | 66.0% |
| Fast Track & Microwaves | 26.0% | 66.0% |
| Fiction & Tamil | 40.0% | 100% |
| Fiction & Timex | 33.0% | 83.0% |
| Microwaves & Tamil | 20.0% | 60.0% |
| Tamil & Timex | 33.0% | 55.0% |

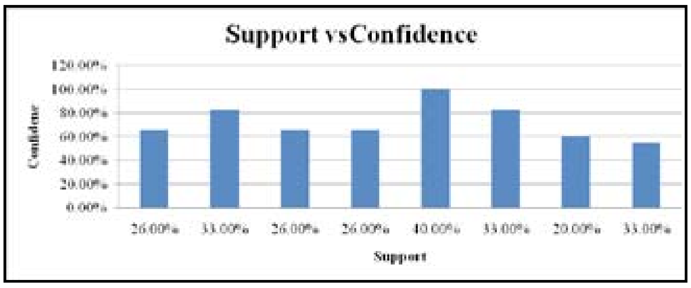


Fig. 3: Support Vs Confidence

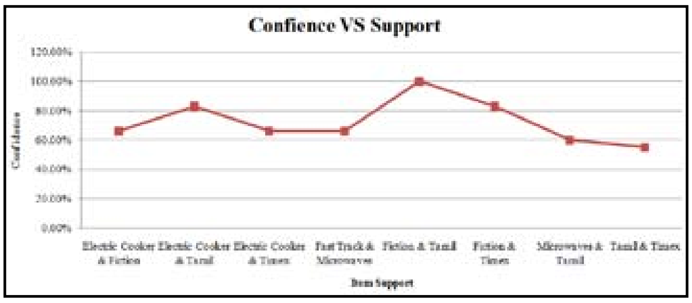


Fig. 4: Support Vs Confidence (with Item Details)

Table 2: Index for Data Retrieval

| ITEMS | SUPPORT | CONFIDENCE |
|---|---|---|
| Electric Cooker, Fiction & Tamil | 26.0% | 100% |
| Electric Cooker, Fiction & Timex | 26.0% | 100% |
| Electric Cooker, Tamil & Timex | 26.0% | 100% |
| Fiction, Tamil & Timex | 26.0% | 80.0% |

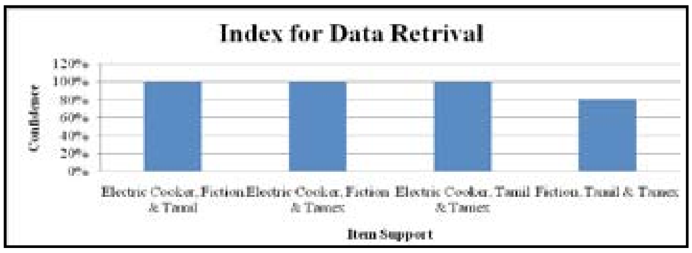


Fig. 5: Index for Retrieval

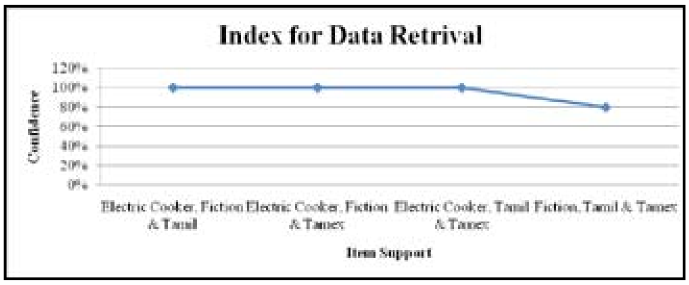


Fig. 6: Index for Retrieval(with Item Details)

Table 3: Search Table info

| Items | Requested | Items | Requested |
|---|---|---|---|
| Fast Track | 29 | English | 2 |
| Titan | 9 | Children | 3 |
| Food Processor | 1 | Kannadam | 2 |
| Hindi | 1 | Cookery | 5 |
| Sonata | 1 | Microwaves | 2 |
| Non Fiction | 2 | Toaster | 1 |
| Citizen | 3 | Malayalam | 1 |

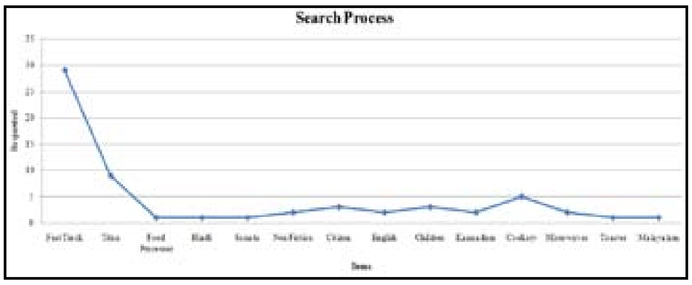


Fig. 7:

## VIII. Conclusion

A flexible technique for index selection is introduced, which can be tuned to achieve different levels of constraints and analysis complexity. A low-constraint more complex analysis can lead to more accurate index selection over stable query patterns. A control feedback technique is introduced for measuring the performance and indicating when the database system could benefit from an index change. The proposed technique affords the opportunity to adjust indexes to new query patterns. The proposed online change detection system utilized two control feedback loops in order to differentiate between inexpensive and more time consuming system changes. In future we reduce the no of computations while computing the index and computing the performance of data retrieval.

## References

[1] G. Valentin, M. Zuliani, D. Zilio, G. Lohman, A. Skelley,"DB2 Advisor: An Optimizer Smart Enough to Recommend Its Own Indexes", Proc. 16th Int'l Conf. Data Eng. 2000.

[2] S. Kai-Uwe, E. Schallehn, I. Geist,"Autonomous Query-Driven Index Tuning", Proc. Fourth Int'l Database Eng. And Applications Symp., 2004.

[3] R.L.D.C. Costa, S. Lifschitz,"Index Self-Tuning with Agent-Based Databases", Proc. 28th Latin-Am. Conf. Informatics, 2002.

[4] S. Berchtold, D. Keim, H. Kriegel,"The X-Tree: An Index Structure for High-Dimensional Data", Proc. 22nd Int'l Conf. Very Large Data Bases, pp. 28-39, 1996.

[5] C.-W. Chung, G.-H. Cha,"The GC-Tree: A High-Dimensional Index Structure for Similarity Search in Image Databases", IEEE Trans. Multimedia, Vol. 4, No. 2, pp. 235-247, June 2002.

[6] N. Bruno, S. Chaudhuri,"To Tune or Not to Tune? A Lightweight Physical Design Alerter", Proc. 32nd Int'l Conf. Very Large Data Bases, pp. 499-510, 2006.

[7] J. Han, J. Pei, Y. Yin,"Mining Frequent Patterns without Candidate Generation", Proc. ACM SIGMOD Int'l Conf. Management of Data, pp. 1-12, 2000.

[8] J. Pei, J. Han, R. Mao,"CLOSET: An Efficient Algorithm for Mining Frequent Closed Itemsets", Proc. ACM SIGMOD Workshop Research Issues in Data Mining and Knowledge Discovery , pp. 21- 30, 2000.

[9] Capara, M. Fischetti, D. Maio,"Exact and Approximate Algorithms for the Index Selection Problem in Physical Database Design", IEEE Trans. Knowledge and Data Eng., 1995.

[10] S. Chaudhuri, V. Narasayya,"AutoAdmin 'What-If' Index Analysis Utility", Proc. ACM SIGMOD Int'l Conf. Management of Data , pp. 367-378, 1998.

[11] S. Chaudhuri, V.R. Narasayya,"An Efficient Cost-Driven Index Selection Tool for Microsoft SQL Server", VLDB J., pp. 146-155, 1997.

[12] S. Agrawal, S. Chaudhuri, L. Kolla´ r, A.P. Marathe, V.R. Narasayya, M. Syamala,"Database Tuning Advisor forMicrosoft SQL Server 2005", Proc. 30th Int'l Conf. Very Large Data Bases, pp. 1110-1121, 2004.

[13] Dogac, A.Y. Erisik, A. Ikinci,"An Automated Index Selection Tool for Oracle7: Maestro 7", Technical Report LBNL/PUB-3161, Software Research and Development Center, Scientific and Technical Research Council of Turkey, 1994.

[14] S. Ponce, P.M. Vila, R. Hersch,"Indexing and Selection of Data Items in Huge Data Sets by Constructing and Accessing Tag Collections", Proc. of Mass Storage Systems and Technologies, 2002.