DATA DISSEMINATION AND DATA SHARING
ANONYMOUS DATABASE BY PROVIDING DATA
SECURITY

M. Sailaja
1. Asst prof , cse.Avanthi college of engineering and technology,Narsipatnam.

Abstract : This paper presents the Btree index, a general and compact structure which provides tight integration of item set extraction in a relational DBMS. The Data bases may be Transactional Data bases or Relational Data bases. Since no constraint is enforced during the index creation phase, Btree index provides a complete representation of the original database. The Btree Index creates prefix tree like structure and stores addresses of disk blocks where relational data base records are stored. Along with the disk blocks Btree index maintains pointers to disk blocks. Using these pointers it extracts required data base records from data base into main memory and finds corresponding item sets for association rule mining. To reduce the I/O cost, data accessed together during the same extraction phase are clustered on the same disk block. The Btree index structure can be efficiently exploited by different item set extraction algorithms. In particular, Btree index data access methods currently support the FP-growth, IBTree, FPMax and LCM v.2 algorithms, but they can straightforwardly support the enforcement of various constraint categories. The Btree index has been integrated into the SQL SERVER, PostgreSQL, Oracle DBMS and exploits its physical level access methods. Experiments, run for both sparse and dense data distributions, show the efficiency of the proposed index and its linear scalability also for large data sets. Item set mining supported by the Btree index shows performance always comparable with, and often (especially for low supports) better than, state-of-the-art algorithms accessing data on flat file.

1. INTRODUCTION

In Today’s Business environment, it is very desirable for larger Business organizations to know the customer interest in hasing the items. This is particularly useful in Market Basket analysis. The Association rules using Item Set data are useful in Business Analysis, Business Intelligence, Business Strategy, Market Basket Analysis, Strategic Planning, and Catalog Design. Association rule mining discovers correlations among data items in a transactional database D. Each transaction in D is a set of data items. Association rules are usually represented in the form A \rightarrow B, where A and B are item sets, i.e., sets of data Items. Item sets are characterized by their frequency of occurrence in D, which is called support. Research activity usually focuses on defining efficient Algorithms for item set extraction, which represents the most computationally intensive knowledge extraction task in association rule mining [1]. The data to be extracted and analyzed using data mining is usually stored into binary files, possibly extracted from a DBMS. Item Set Mining involves three phases. 1) Data Extraction 2) Item Set Preparation 3) Finding Association Rules. Most Algorithms identified from [1],[2],[3],[4],[5],[6] exploit ad hoc main memory data structures to efficiently extract item sets from a flat file. Recently, disk-based extraction algorithms [7],[8],[9] have been proposed to support the extraction from large data sets but still dealing with data stored in flat files. To reduce the computational cost of item set extraction, different constraints maybe enforced among which the most simple is the support constraint, which enforces a threshold on the minimum support of the extracted item sets. Algorithms identified from [1],[2],[3],[4],[5],[6] produces limitations such as If the query patterns change it does not provide better result, Query response does not perform well if query patterns changes. i.e speed of execution, Efficiency and reliability is less, performance may degrade if the database size gets increases. Tradition feature selection technique may offer less or no data pruning capability given query attributes, Query response does not perform well if query patterns changes and performance may degrade if the database size gets increased.

1.1 Characteristics of Item Set Mining using BTree Index:

Relational DBMS’s exploit indices, which are ad-hoc data structures to enhance query performance and support the execution of complex queries. In this paper, a similar approach to data mining queries is presented. The
Association Rule mining on relational data base is proposed to perform Item Set mining efficiently and extract item sets from relational data base. The BTree index is a novel data structure that provides a compact and complete representation of transactional data supporting efficient item set extraction from a relational DBMS. It is characterized by the following properties:

1. It is a covering index.

That means, nothing like support constraint is enforced during the Index creation phase. Because of this reason BTree supports to perform Queries of different Patterns. The Data Extraction algorithms can extract data items without accessing the original data base.

2. It is incremental index also called as I-Forest Index. A technique for the incremental update of the index, suitable for association rule extraction on evolving databases, has been proposed and implemented. By means of the proposed index called I-Forest, incoming data blocks are stored on disk in appropriate (compact) structures. During the mining phase, only the data required by the current mining process is actually loaded in main memory. To allow different kinds of analysis and easy incremental insertion of new blocks (or deletion of obsolete ones), each data block is represented separately and independently of all others.

3. A control feedback technique is introduced for measuring the performance.

4. Through this a database could benefit from an index change. The index selection minimizes the cost of the queries in the work load.

5. Online index selection is designed in the motivation if the query pattern changes over time.

6. BTree supports the enforcement of various constraint categories.

7. The Btree Physical organization supports efficient data access during item set extraction. Correlation analysis is used for data access together during pattern extraction. This minimizes number of physical data blocks read during the Mining process.

This paper is organized as follows: Section 2 completely describes The Item Set Mining using Btree index System Design and Implementation. Section 3 describes Btree index architecture and work flow by addressing its components. Section 4 describes data access methods. Section 5 describes Btree index structure, by addressing its structure, its data access methods, and its physical organization. Section 6 describes how the IB-Tree and LCM v.2 algorithms may exploit Btree index to perform efficiently the extraction of item sets. It also describes how the Btree index supports the enforcement of various constraint types. Section 7 presents description about experiment of Btree index. Section 8 presents conclusion and future scope.

2. System Design and Implementation:

The overall system design of Item Set Mining using BTree Index is described in Figure 1.

![Figure 1. Layered System Design](image)

Item Set Mining using BTree Index is composed of 5 layers.

1) The User Interface Layer:

It is responsible for interaction with the user and various calls to various graphical and visualization utilities. This Module provides an Interface for each user to invoke with the system and to execute Queries for generating Frequent Item Sets (Frequent Item Patterns). It provides the following services

1.1) To design interface for IBtree algorithm to generate frequent patterns. The Interface receives parameters Minimum support and Minimum confidence called Interesting measures (interesting constraints) and produces frequent patterns as output.
1.2) To design interface for LCM algorithm to generate frequent patterns. The Interface receives parameters Minimum support and Minimum confidence called Interesting measures(interesting constraints) and produces frequent patterns as output.

1.3) An Interface to displays the performance results of IBTree and LCM algorithms.

2). Data Mining Algorithms Layer: This is the main module connecting all system components. In this module there are set of Data Mining Algorithms to execute Query and generate frequent Item Sets. Here there are two algorithms to generate frequent item sets for finding Association rules among them.

1)IBTree Algorithm
2) LCM Algorithm

2.1) IBTree Algorithm:
It receives set of transactional data items from relational data model and two interesting measures Min Support, Min Confidence as input, and then it constructs an FPTree for indexing and generates frequent item sets. During this process it communicates with Index Layer which constructs Index for extracting Items from Data base.

2.2) LCM Algorithm:
This is Linear Time Closed Item Set Mining. It receives set of transactional data items from relational data model, two interesting measures Min Support, Min Confidence and then generates Frequent Item Sets with the help of BTtree Index at Index Layer. During the process of generating Frequent Item Sets, it uses array based structure than tree structure.

3) Index Layer:
This Module mainly focuses on creating index for data items located in data base which is used by Data Mining algorithms to extract data items in faster way for generating frequent item sets. Here the Index is Incremental Index and it is constructed using BTree and ITree data structures and it is called as BTree Index.

4) Data Base Server Layer: SQL Server relational data base. Contains data base of transactional data items. Receives Queries from the user interface design and transfers results from transactional data base.

5) Data Base Layer: Currently available in the proposed system are Transactional data about market basket analysis.

3. Btree Index Architecture and Flow: The general flow and architecture of Btree index is described in figure 2 and figure 3.

System Architecture-I (Level-I)

Figure 2. Work Flow

System Architecture-II (Level-II)

Figure 3. System Architecture as Two components

Architecture of Item Set Mining using BTree Indexing is divided into two parts

1.) BTree Index Structure.
2.) Data access Methods.

The BTree index actually organizes addresses of disk blocks where data items regarding relational databases or Transactional databases are stored in disk in such a way that data access methods can directly read required data into memory to perform item set mining. More over they read only required data into memory. The structure of BTree index is characterized by two components.

1) The Item set-Tree
2) The Item B-tree

The two components are responsible for physical organization of database data. The two components provide two levels of indexing. The item set tree also called I-tree is a prefix tree which represents relation R as less compact structure it represents addresses of disk blocks in the form of tree according to item sets involved in association rule mining. The item set-tree can be based on different kinds of trees like FP-tree, H-tree, Inverted Matrix, Patricia-trees.
4) BTree Data Access Methods:
The BTree index structure is independent of the item set
Extraction algorithm and Association rule mining
algorithm. For this reason different state of art algorithms
(different item set extraction algorithms) may be used in
support of BTree index. The algorithms like FPtree for
Fpgrowth, IB tree algorithm, array based structure for
LCM can be used in support of BTree index to perform
item set extraction and association rule mining.
Depending on support and data items chosen in algorithm
a different portion of I-tree is accessed for reading data
item from the disk.

5) Btree Index Structure:
The transactional data set D is represented, in the
relational model, as a relation R. Each tuple in R is a pair
(TransactionID, ItemID). The Btree index provides a
compact and complete representation of R. It is two level
indexing and incremental indexing. Hence, it allows the
efficient extraction of item sets from R, possibly
enforcing support or other constraints. In Section 5.1, the
general structure of the BTree index is presented; while in
Section 5.2, how data access takes place is discussed. The
physical organization of the index is presented in Section
5.3 together with a discussion of access cost.

5.1 Btree Index structure:
The structure of the Btree index is characterized by two
components: the Item set-Tree and the Item-Btree. The
two components provide two levels of indexing. The Item
set-Tree (I-Tree) is a prefix-tree which represents relation
R by means of a succinct and lossless compact structure.
The Item-Btree (I-Btree) is a B+Tree structure which
allows reading selected I-Tree portions during the
extraction task. For each item, it stores the physical
locations of all item occurrences in the I-Tree. Thus, it
supports efficiently loading from the I-Tree the
transactions in R including the item. In the following
section details of the I-Tree and the I-Btree structures is
described.

5.1.1 I-Tree
An effective way to compactly store transactional records
is to use a prefix-tree. Trees and prefix-trees have been
frequently used in data mining and data warehousing
indices, including cube forest, FP-tree [3], H-tree,
Inverted Matrix [7], and Patricia-Tries. The current
implementation of I Tree is based on FP Tree and it is
supported to be efficiently accessed by IB Tree and LCM
algorithms.

The I-Tree associated to relation R is actually a forest of
prefix-trees, where each tree represents a group of
transactions all sharing one or more items. Each node in
the I-Tree corresponds to an item in R. Each path in the I-
Tree is an ordered sequence of nodes and represents one
or more transactions in R. Each item in relation R is
associated to one or more I-Tree nodes and each
transaction in R is represented by a unique I-Tree path.

The Figure 4 shows a small data set and Figure 5 shows
complete structure of BTree Index. In the I-Tree paths (Fig.
5a), nodes are sorted by decreasing support of the
Corresponding items. In the case of items with the same
support, nodes are sorted by item alphabetical order. Each
I-Tree node is associated with a node support
value, representing the number of transactions which
contain all the items in the sub path reaching the node. For
example, in sub path [e:3, h:3]; the support of node [h:3]
is 3.
Hence, this sub path represents three transactions (i.e., transactions 3, 4, and 9). Each item is associated to one or more nodes. The item support is obtained by adding the support of all nodes including the item Nodes in the I-Tree are linked by means of pointers which allow selectively loading from disk the of the corresponding node.

5.1.2 I-Btree
The I-Btree allows selectively accessing the I-Tree disk blocks during the extraction process. It is based on a B+Tree structure. Fig. 5b shows the I-Btree for the example data set and a portion of the pointed I-Tree. For each item i in relation R, there is one entry in the I-Btree. In particular, the I-Btree leaf associated to i contains i's item support and pointers to all nodes in the I-Tree associated to item i.

5.2 Data Access Methods
BTree Index is general index structure. Hence, different data mining algorithms may be applied, once data is loaded into memory. The in-memory representation is suitable for the selected extraction algorithm such as FP-tree for FP-growth, IBTree for IBTree algorithm, array-based structure for LCM.

5.2.1 Loading the Frequent-Item Projected Database
The frequent-item projection of relation R with respect to an arbitrary item Ĵ includes the transactions in R where Ĵ occurs, intersected with the items having higher support than Ĵ or equal support but preceding Ĵ in lexicographical order. In the I-Tree paths, items are sorted by descending support and lexicographical order. Thus, the projection is represented by the I-Tree prefix paths of item Ĵ.

5.2.2 Loading the Support-Based Projected Database
The support-based projection of relation R contains all transactions in R intersected with the items which are frequent with respect to a given support threshold MinSup. The I-Tree paths completely represent the database transactions. Items are sorted by decreasing support along the paths. Thus, the support-based projection of R is given by the I-Tree sub paths between the I-Tree roots and the first node with an unfrequented item.

Figure 6. Loading the Frequent-Item Projected Database

Figure 7. Loading the Support-Based Projected Database

5.3 Physical Organization
The physical organization of the BTree index is designed to minimize the cost of reading the data needed for the current extraction process. The I-Btree allows a selective access to the I-Tree paths of interest. Hence, the I/O cost is mainly given by the number of disk blocks read to load the required I-Tree paths. The I-Tree physical organization is based on the following correlation types:

1) Intratransaction correlation.
   Extraction algorithms consider items occurring in the same transaction. Items appearing in a transaction are thus intrinsically correlated.

2) Intertransaction correlation.
   Transactions with some items in common will be accessed together when item sets including the common items are extracted.

5.3.1 I-Tree Layers
I-Tree paths are partitioned into three layers, based on node access frequency.

Top Layer. This layer includes nodes that are very frequently accessed during the mining process. These nodes are located in the upper levels of the I-Tree. They correspond to items with high support, which are distributed over few nodes with high node support.

Middle Layer. This layer includes nodes that are quite frequently accessed during the mining process. These nodes are typically located in the central part of the tree. They correspond to items with relatively high support, but not yet dispersed on a large number of nodes with very low node support.

Bottom Layer. This layer includes the nodes corresponding to rather low support items, which are rarely accessed during the mining process. Nodes in this...
layer are analyzed only when mining frequent item sets for very low support thresholds

6. The IB-Tree and LCM v.2 algorithms

6.1 LCM Algorithm.
LCM is an abbreviation of Linear Time Closed item set miner. This is used for enumerating all frequent item sets. In LCM we define parent-child relation ship between frequent closed item sets.

Algorithm LCM (X: frequent closed item set)

1. output X
2. For each i>i(X) do
3. if X[i] is frequent and X[i] = I(T(X[i])) then
4. Call LCM(X[i])
5. end for

The following is detailed description of algorithm. Global T,DT /* Global Lists */

Algorithm LCM(

1. X:=I(T(ș)) /* The Root */
2. For i:=1 to |E|
3. if(X[i]) satisfies (cond2) and (cond3) THEN
4. Call LCM_Iter(X[i],T(X[i]),i) or
5. Call LCM_Iter2(X[i],T(X[i]),I,DT)
6. end for

Algorithm LCM_Iter(X,T(X),i(X))

1. Output X
2. for each TřT(X)
3. for each ķřT(X), Insert I to T(ķ)
4. for each ĵ, T(ķ) ≠ ș in the decreasing order
5. if(|T|į)≻J and (cond2) holds then
6. LCM_Iter(T(ķ),T(į),į)
7. delete T(į)
8. end for

7. EXPERIMENTAL RESULTS
The proposed approach is validated by means of a large set of experiments addressing the following issues:
1. Performance of the Btree index creation, in terms of both creation time and index size.
2. Performance of frequent item set extraction, in terms of execution time, memory usage.

Figure 8.1, 8.2 describes the results (Frequent Item Sets) of IBTree Algorithm

Figure 8.1 Results of IBTree Figure 8.2 Results of IBTree

Figure 9.1, 9.2, describes the results (Frequent Item Sets) of LCM Algorithm

Figure 9.1 Results of LCM Figure 9.2 Results of LCM

Figure 10 shows the performance results of IBTree Algorithm and LCM algorithm (Support and Time Analysis) and Figure11 shows the performance results of IBTree Algorithm and LCM algorithm (Support and Memory Size Analysis)

Figure 10. Time complexity Figure 11. Space Complexity Figure12 shows Support and time analysis of IBTree Algorithm and Figure 13 shows Support and time analysis of IBTree Algorithm.

Figure 12. Time complexity Figure 13. Time complexity

Figure 14 shows Support and memory size analysis of IBTree Algorithm and Figure 15 shows Support and memory size analysis of LCM Algorithm.
8. CONCLUSIONS AND FUTURE WORK

The BTree index is a novel index structure that supports efficient item set mining into a relational DBMS. It has been implemented into the PostgreSQL open source DBMS, SQL Server, Oracle by exploiting its physical level access methods. The BTree index provides a complete and compact representation of transactional data. It is a general structure that efficiently supports different algorithmic approaches to item set extraction. Selective access of the physical index blocks significantly reduces the I/O costs and efficiently exploits DBMS buffer management strategies. This approach, implemented into a relational DBMS, yields better performance than the state-of-the-art algorithms (i.e., Prefix-Tree and LCM v.2 accessing data on a flat file and is characterized by a linear scalability also for large data sets with incremental update of the index. When the transactional database is updated, the BTree index needs not to be rematerialized. It incrementally updates the index when new data become available. Since no support threshold is enforced during the index creation phase, the incremental update is feasible without accessing the original transactional database.

REFERENCES


