P-3559

PERRFORMANCE ANALYSIS OF INDEXING TECHNIQUES IN DATAWAREHOUSING

PROJECT REPORT

Submitted by

B.DIVIYAPRIYA

Reg. No: 0920108003

In partial fulfillment for the award of the degree

of

MASTER OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY

(An Autonomous Institution Affiliated to Anna University of Technology, Coimbatore)

COIMBATORE – 641 049

APRIL 2011

KUMARAGURU COLLEGE OF TECHNOLOGY

(An Autonomous Institution Affiliated to Anna University of Technology, Coimbatore)

COIMBATORE – 641 049

Department of Computer Science and Engineering

PROJECT WORK

PHASE II

APRIL 2011

This is to certify that the project entitled

PERFORMANCE ANALYSIS OF INDEXING TECHNIQUES IN DATAWAREHOUSING

is the bonafide record of project work done by

B.DIVIYAPRIYA

Register No: 0920108003

of M.E. (Computer Science and Engineering) during the year 2010-2011.

[illegible]
Project Guide

[illegible]
Head of the Department

Submitted for the Project Viva-Voce examination held on 25-4-2011

[illegible] 25/4/11
Internal Examiner

[illegible] 25/4/2011
External Examiner

DECLARATION

I affirm that the project work titled **PERFORMANCE ANALYSIS OF INDEXING TECHNIQUES IN DATAWAREHOUSING** being submitted in partial fulfillment for the award of M.E. degree is the original work carried out by me. It has not formed the part of any other project work submitted for the award of any degree or diploma, either in this or any other University.

[illegible]
B.DIVIYAPRIYA

Register No: 0920108003

I certify that the declaration made above by the candidate is true

[illegible]

Signature of the Guide

Mr.K.R.BASKARAN M.E.,

Associate Professor

Department of Information Technology,
Kumaraguru College of Technology,
(An Autonomous Institution)
Coimbatore-641 049.

ACKNOWLEDGEMENT

First and foremost, I would like to thank the Lord Almighty for enabling me to complete this project.

I express my profound gratitude to our Chairman **Padmabhusan Arutselvar Dr.N.Mahalingam, B.Sc., F.I.E.,** for giving this opportunity to pursue this course.

I would like to thank **Dr.S.Ramachandran,Ph.D.,** *Principal* for providing the necessary facilities to complete my thesis.

I take this opportunity to thank **Mrs.P.Devaki M.E.., (Ph.D..,),** *HOD* Department of Computer Science and Engineering, for her precious suggestions.

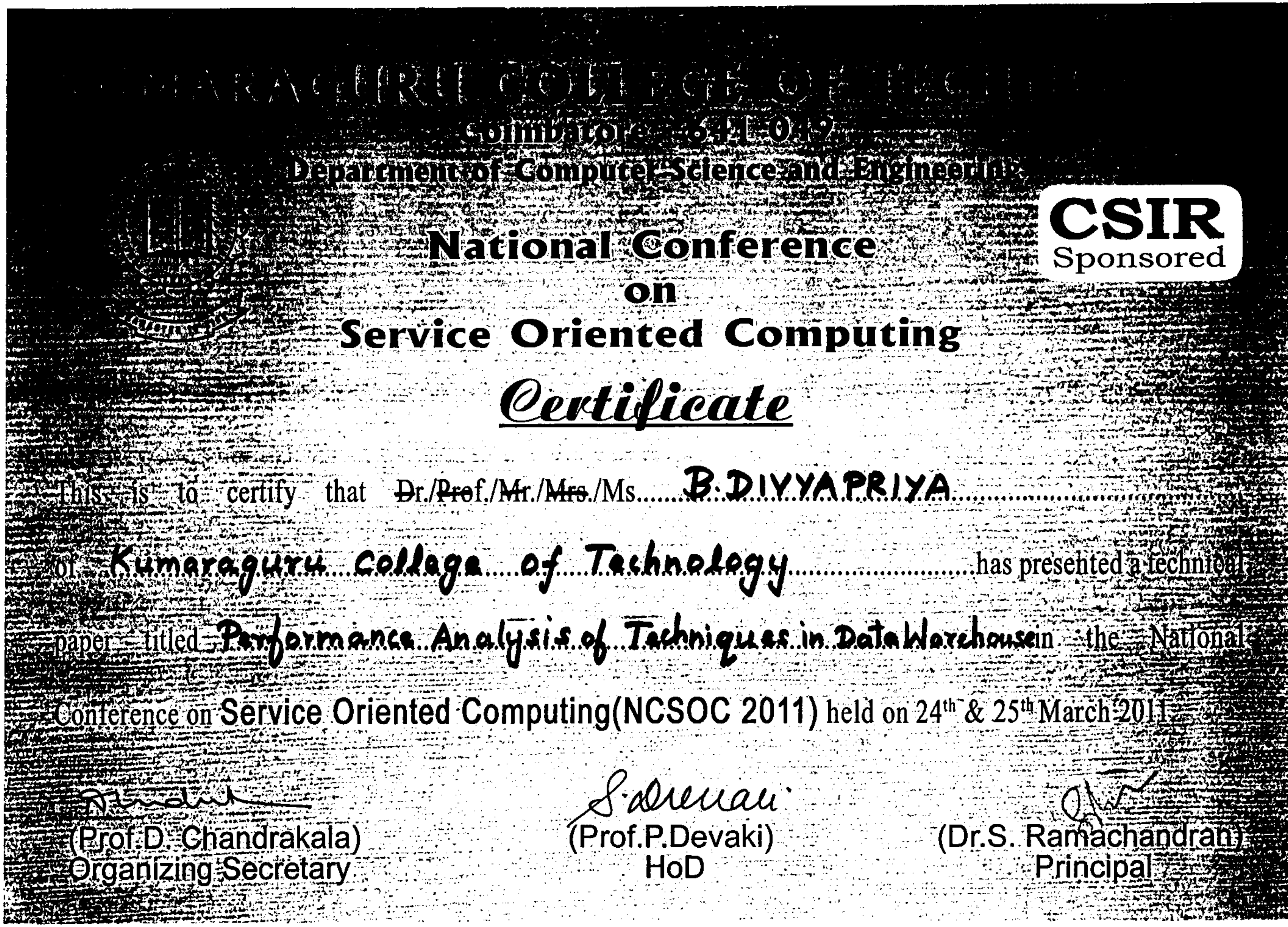
I thank all project committee members for their comments and advice during the reviews. Special thanks to **Mrs.V.Vanitha M.E,** *Associate Professor and* **Mr.V.Subramani M.Tech,** *Associate Professor*, Department of Computer science and Engineering, for arranging the brain storming project review sessions.

I register my hearty appreciation to the Guide **Mr.K.R.Baskaran M.E.,** *Associate Professor,* Department of Information Technology, my thesis advisor. I thank for his support, encouragement and ideas. I thank him for the countless hours he has spent with me, discussing everything from research to academic choices.

I would like to convey my honest thanks to all **Teaching** staff members and **Non Teaching** staffs of the department for their support. I would like to thank all my classmates who gave me a proper light moments and study breaks apart from extending some technical support whenever I needed them most.

I dedicate this project work to my **parents** for no reasons but feeling from bottom of my heart, without their love this work wouldn't possible.

TABLE OF CONTENTS



KUMARAGURU COLLEGE OF TECHNOLOGY

Coimbatore - 641 049

Department of Computer Science and Engineering

CSIR Sponsored

National Conference on Service Oriented Computing

Certificate

This is to certify that ~~Dr.~~/~~Prof.~~/~~Mr.~~/~~Mrs.~~/Ms. B. DIVYAPRIYA of Kumaraguru College of Technology has presented a technical paper titled Performance Analysis of Techniques in Data Warehouse in the National Conference on Service Oriented Computing(NCSOC 2011) held on 24th & 25th March 2011.

[illegible]
(Prof.D. Chandrakala)
Organizing Secretary

[illegible]
(Prof.P.Devaki)
HoD

[illegible]
(Dr.S. Ramachandran)
Principal

ABSTRACT

The explosive growth of information in recent years has brought up a new challenge of disseminating information from a huge plethora of data. The data warehousing is a large repository .It is mainly used for decision making. But majority of requests for information from data warehouse involves ad -hoc queries. The ability to answer them is a very critical issue. There are many techniques to speed up the query processing such as summary tables, but one among them is indexing technique. The implementation of three indexing techniques namely cluster, hash, bitmap indexing are carried out and the comparison of their performance is studied. In this proposed system in order to speed up the query processing using bitmap index method an enhancement is performed using simple bitmap techniques, range based techniques, bit sliced indexing techniques and projection indexing techniques.

ஆய்வுச்சுருக்கம்

சமீப காலத்தில் தகவல்களின் அதிக அளவிலான வளர்ச்சியால் தகவல்களை அதிகப்படியான தரவுகளில் இருந்து பரவச்செய்தல் என்பது ஒரு புதிய சவாலாக உள்ளது.தரவுக்கிடங்கு என்பது ஒரு மிகப்பெரிய களஞ்சியம்.இத்தரவுக்கிடங்குகளின் உதவியால் முக்கியமான முடிவுகள் எடுக்கப்படுகின்றன. ஆனால் பெரும்பாலான குறித்த காரியத்திற்காக அமைக்கப்பட்ட வினாக்கள் தரவுக்கிடங்குகளில் இருந்து எடுக்கப்படுகின்றன. ஆனால் பெரும்பாலான குறித்த காரியத்திற்காக அமைக்கப்பட்ட வினாக்கள் தரவுகிடங்குகளில்யிருந்து எடுக்கப்படுகிறது. அந்த வினாக்களுக்கு விடையளிக்கும் திறன்னானது மிகப் பெரிய இக்கட்டான நிலை.பெரும்பாலான முறைகள் இந்த வினாசெயல்பாடினை வேகபடுத்துவதற்காக உள்ளன. அவற்றில் ஒன்று குறியிடுதல்.அமல்படுத்தப்பட்ட மூன்று வகையான குறியீடுகள் முறையே தொகுப்பு குறியிடுதல்,வெட்டு குறியிடுதல்,துண்டுவரைப்பட குறியிடுதல் மற்றும் அதன் திறமைகள் அனைத்தும் ஒப்பிடப்பட்டுள்ளது.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

DW –Datawarehousing

OLAP-Online Analytical Processing

OLTP-Online Transaction Processing

JSCE-Japan Society of Civil Engineers

SVM-Support Vector Machine

CHAPTER 1
INTRODUCTION

1.1 OVERVIEW OF DATA WAREHOUSING

The data warehouse is a collection of large volume of data from multiple sources. It is a data repository. The data warehouse is mainly for decision making. Data warehousing is a integrated, subject-oriented, Time-variant and is primarily used for organizational decision making. A Data Warehouse (DW) is a collection of technologies and techniques which assists the following knowledge workers:

- Executive
- Manager
- Analyst

To make better and faster decisions using various analytical tools. Data warehouse is a type of computer database that is responsible for collecting and storing the information of a particular organization. The goal of using a data warehouse is to have an efficient way of managing information and analyzing data. A data warehouse (DW) is a database used for reporting. The data is off loaded from the operational systems for reporting. A data warehouse maintains its functions in three layers: staging, integration and access. A principle in data warehousing is that there is a place for each needed function in the DW. The functions are in the DW to meet the users reporting needs. The main source of the data is cleaned, transformed, catalogued and made available for managers and other business professionals for data mining, online analytical processing, market research and decision support many references to data warehousing use this broader context. Thus, an expanded definition for data warehousing includes business intelligence tools, tools to extract, transform and load data into the repository, and tools to manage and retrieve metadata.

Architecture, in the context of an organization's data warehousing efforts, is a conceptualization of how the data warehouse is built. There is no right or wrong architecture, but rather there are multiple architectures that exist to support various environments and situations. The worthiness of the architecture can be judged from how the conceptualization aids in the building, maintenance, and usage of the data warehouse.One possible simple conceptualization of data warehouse architecture consists of the following interconnected layers:

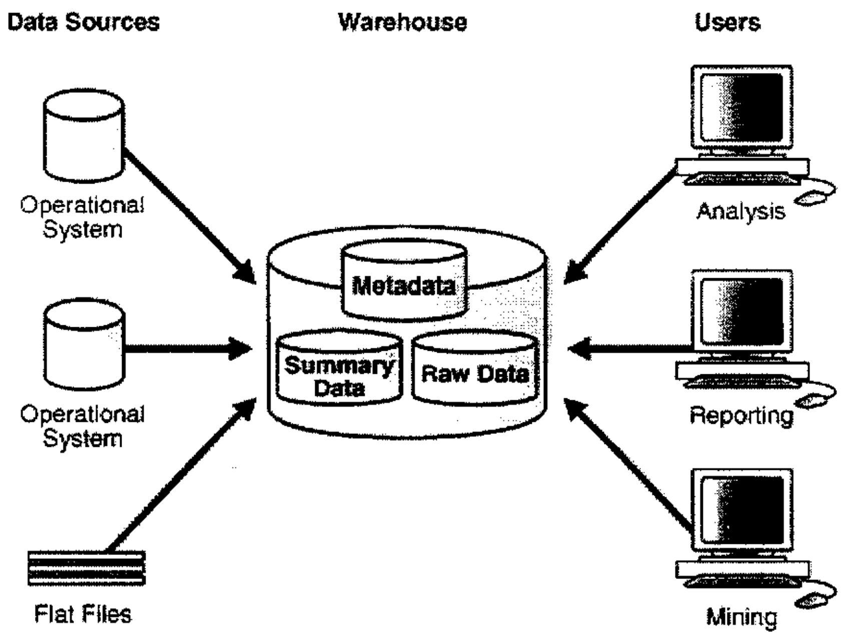


Figure1.1 Architecture of a Data Warehouse

1.1.1 NEED FOR DATAWAREHOUSING:

- To make better and faster decisions using various analytical tools. Data warehouse is a type of computer database that is responsible for collecting and storing the information of a particular organization.
- The goal of using a data warehouse is to have an efficient way of managing information and analyzing data.

- A data warehouse (DW) is a database used for reporting. The data is off loaded from the operational systems for reporting. Enhances end-user access to a wide variety of data.
- Business decision makers can obtain various kinds of trend reports e.g. the item with the most sales in a particular area / country for the last two years.
- Creation of a computing infrastructure that can support changes in computer systems and business structures.
- Empowering end-users to perform any level of ad-hoc queries or reports without impacting the performance of the operational systems.

1.1.2 CHARECTERISTICS OF INDEXING TECHNIQUES:

- Support ad hoc and complex queries.
- Speed up join operations.
- Easy to build.
- Easy to implement and maintain.
- Should occupy less memory space.
- Utilize space efficiently.

CHAPTER 2

LITERATURE SURVEY

- Index selection techniques for data warehousing.
- Techniques used for Processing Datawarehouse queries.
- A method of cluster-based indexing for textual data.
- Hash-based indexes.
- Bitmap index.

2.1 INDEX SELECTION TECHNIQUES FOR DATAWAREHOUSING

2.1.1 DESCRIPTION:

To achieve this goal a rule-based optimizer is used which generates the query execution plans that are evaluated according to specific cost model. The indexes considered belong to two very common categories : tid - list indexes and bitmap indexes ,both access a B+-tree. Finally a greedy algorithm is used which chooses from a set of candidate indexes .The validity of the approach is evaluated with reference to some experimental tests, part of which aimed to comparing the relative benefits arising from view materialization and indexing.A basic requirement of the DW user is to obtain quick queries answer for their queries.On of the most effective to achieve this goal during logical design is view materialization. A view contains aggregated data obtained from the fact table containing elemental data. The aggregation level characterizing the view is called Pattern and consist of a set of star schema .Designing the physical schema requires to determine the set of index to built both on fact tables and dimensional table.In this, a heuristical approach to index selection in data warehouse is achieved through the star schema. In the logical schema an OLAP workload, data volume, and a constraint defining the disk space to access the goal is to determine the optimal physical schema, to reduce the workload cost. For this purpose the rule-based optimizer model are used which is capable of execution plan for each query.Here an architectural sketch is made on which this approach work

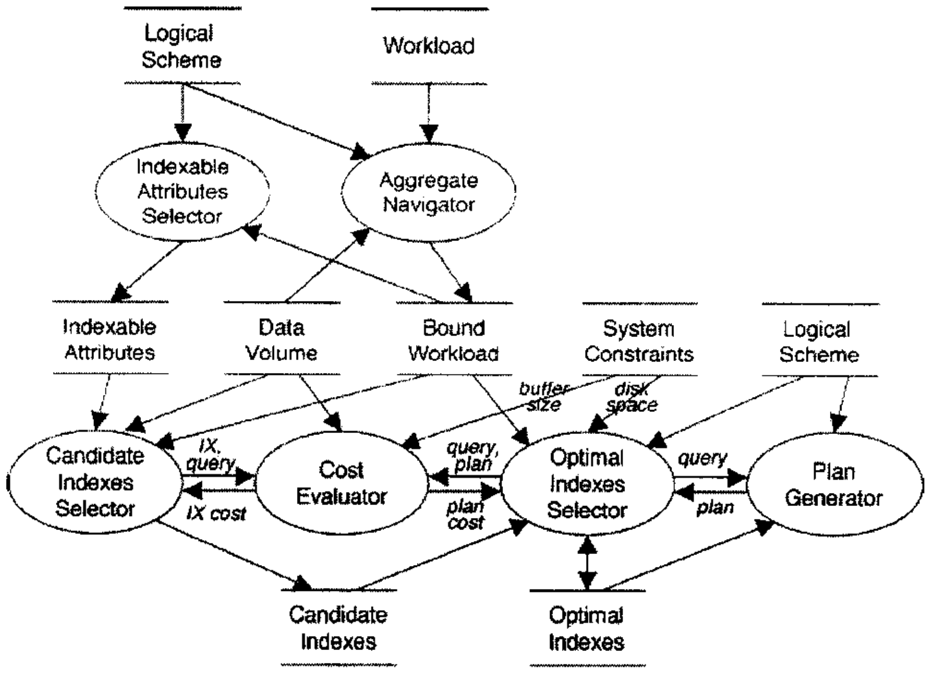


Figure: 2.1 Architectural sketch

The Logical schema is the relational schema for fact tables. The workload is a set of queries executed on the DW. The data volume contains quantitative information about data. System constraints include available disk space. The bound workload which couples each query in the work load. The aggregate navigator it has one or more materialized views. The index table attributes is based on the structure of the queries. The candidate index selection evaluates which type of index is convenient.The cost evaluator to evaluate both the cost of each index and the execution plan.

2.1.1.1 OPERATION OF EXEUTION PLANS:

Table scan sequentially scans a table and returns the set of all tuples. Index scan access an index and retrieves the tids of the tuples that satisfy the selection predicate. Table access a table to get the tuple related to the set of tids of tuples. Hash join carries out the natural join between the two set of tuples.Tid intersection returns the intersection between two sets of tids.

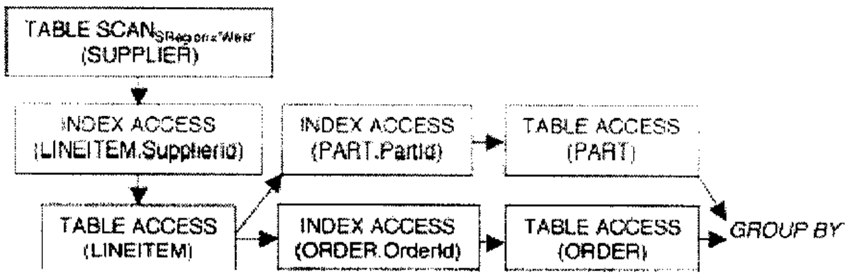


Figure: 2.2 Operation of execution plans

2.1.1.2 INDEX ALGORITHM:

Indexing reduces the workload execution cost. Here the primary keys are created and the cost when beneficial index are created in function of the constraint VS on the disk space available for materialization.The Data warehousing is becoming more and more important for decision-makers. Most of the queries implemented on a large data warehouse are complex and iterative .The ability to retrieve the answers of the query is a critical issue in data warehouse. There are many solutions to speed up the query processing time such as summary table, OLAP and Indexes. Here in this project the focus is on indexes.

Main challenge in data warehouse

Query performance using complex queries.

Solution

Indexing techniques for complex queries.

Indexes

- Database objects associated with database tables
- To speed up the data access time within the tables.

2.1.1.3 ADVANTAGES OF INDEXING TECHNIQUES:

- Faster key-based access to table data
- Reduced storage requirement
- Efficient retrieval

2.1.1.4 DISADVANTAGES OF INDEXING TECHNIQUES:

Few Indexes:

- Data loads up quickly
- Query response time is slow.

Many Indexes:

- Data loads slowly
- Storage space
- Query response is good [7]

2.2 TECHNIQUE USED FOR PROCESSING DATA WAREHOUSE QUERIES

2.2.1 DESCRIPTION

A data warehouse is a large repository of information accessed through an online analytical processing application. This application provides the user with tools to iteratively query the data warehouse in order to make better and faster decision. But majority of requests for information from data warehouse involve ad-hoc queries. The ability to answer this is very critical issue. There are many solutions to speed up query processing such as summary tables, indexes etc. The performance when using summary tables for predetermined queries is good. When an unpredicted query arises the system must scan, fetch and sort the actual data resulting in performance degradation. When the base table changes the summary table has to be changed. Therefore the Indexing technique is the best to achieve this objective without adding any hardware.

2.2.1.1 INDEXING ISSUES IN OLAP AND OLTP

Indexes are databases associated with database table and created to speed up the access to data within the tables. Indexing techniques have already been in existence for decades for OLTP relational database system but they cannot handle a large volume of data and complex queries in OLAP. Because of the following issues the right index structure is to be selected.

Features	OLTP	OLAP
Users	Clerk, IT professional	Knowledge worker(managers,executives,analysts)
Function	Day to day operations	Decision support
DB design	Application-oriented	Subject-oriented
Data	Current,up-to-date detailed, Flat relational isolated	Historical,Summarized, multidimensional integrated, consolidated
Usage	Repetitive	Ad-hoc
Access	Read/write Index/hash on primary key	Lots of scans
Unit of work	Short, simple transaction	Complex query
Records accessed	Tens	Millions
Users	Thousands	Hundreds
DB size	100MB-GB	100GB-TB
Metric	Transaction throughput	Query throughput, response

Table: 2.1 Comparisons between OLAP and OLTP

2.2.1.2 FACTORS TO DETERMINE WHICH INDEXING TECHNIQUES SHOULD BE BUILT ON COLUMN:

- **Cardinality data**

The cardinality is the number of distinct values in the column. The cardinality of indexing column is to be known whether it is low or high, because the indexing techniques works based on the cardinality of column.

- **Distribution**

  The distribution of a column is the occurrence frequency of each distinct value of the column. It is to guide us which indexing should be selected.

- **Value range**

  The range of values of and indexed column guides us to select an appropriate index type.

2.2.1.3 SUPPORT THE INDEXING TECHNIQUES IN THE FOLLOWING BASIS:

- Support ad hoc and complex queries.
- Speed up join operations.
- Easy to build.
- Easy to implement and maintain.
- Should occupy less memory space.
- Utilize space efficiently

2.2.1.4 EVALUATION OF EXISTING INDEXING TECHNIQUE:

Indexing Techniques	Characteristics	Advantages	Disadvantages	Implementing Commercial Systems
B-Tree Index	Two representations (row id and bitmap) are implemented at the leaves of the index depending on the cardinality of the data	· It speeds up known queries. · It is well suited for high cardinality.	· It performs inefficiently with low cardinality data · It does not support ad hoc queries. More I/O operations are needed for a wide range of queries.	· Most of commercial products (Oracle, Informix, Red Brick
Pure Bitmap Index	An array of bits is utilized to represent each unique column value of each roin a table, setting the bits corresponding to the row either ON (valued 1) or OFF (valued 0).	. It is well suited for low cardinality columns. . It utilizes bitwise operations. .	It performs In efficiently with high cardinality data. · It is very expensive when we update index column.. · It does not handle spare data well	· Oracle · Informix · Sybase · Informix · Red Brick · DB2
Encoded Bitmap Index	The index is the binary Bit-Sliced Index built on the attribute domain	·It uses space efficiently. · It performs efficiently with wide range query	. It performs inefficiently with equality queries. . It is very difficult to find a good encoding scheme.	. DB2
Bitmap Join Index	The index is built by restriction of a column on the dimension table in the fact table	· It is flexible. · It performs efficiently. · It supports star queries	. The order of indexed column is important	· Oracle · Informix · Red Brick
Projection Index	The index is built by storing actual values of column(s) of indexed TABLE.	It speeds up the performance when a few columns in the table are retrieved	. It can be used only to retrieve raw data (i.e., column list in selection).	· Sybase

Table 2.2 Comparison of existing indexing techniques [8]

2.3A METHOD OF CLUSTER-BASED INDEXING OF TEXTUAL DATA:

2.3.1 DESCRIPTION:

This paper is an attempt to provide a view of indexing as a process of generating many small clusters overlapping with each other. Individual clusters, referred to as micro clusters in this project, it contain multiple subsets of associated elements, such as documents, terms, authors, keywords, and other related attribute sets. For simplicity,the focus is of primarily on the co-occurrences between 'documents' and 'terms' in our explanation, but the presented framework is directly applicable to more general cases with more than two attributes.

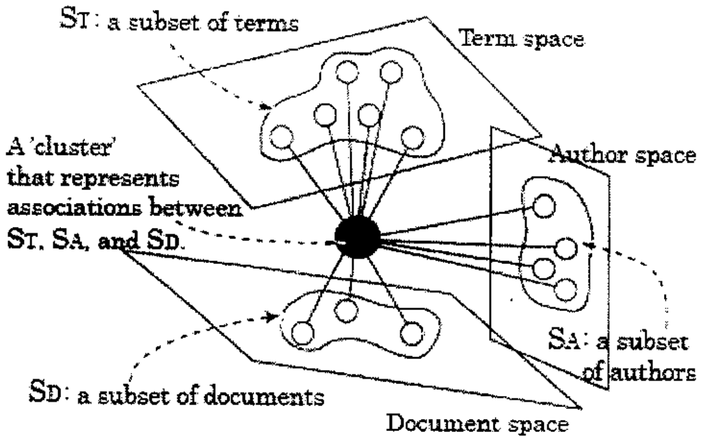

ST : a subset of terms
Term space
A 'cluster' that represents associations between ST, SA, and SD.
Author space
SA : a subset of authors
SD : a subset of documents
Document space

Figure: 2.3 Cluster based indexing techniques of textual data

2.3.1.1 BASIC STRATEGY:

- Instead of generating component vectors with many non-zero elements, produce only limited subsets of elements, i.e., micro-clusters, with significance weights.
- Instead of transforming the entire co-occurrence matrix into a different feature space, extract tightly associated sub-structures of the elements on the graphical representation of the matrix.
- Use entropy-based criteria for cluster evaluation so that the sizes of the generated clusters can be determined independently of other existing clusters.

- Allow the generated clusters to overlap with each other. By assuming that each element
- Can be categorized into multiple clusters, we can reduce the problem to a feasible level where the clusters are processed individually.

2.3.1.2 EXPERIMENTAL RESULTS

In the experiments, the NTCIR-J11, a Japanese text collection for retrieval tasks that is composed of abstracts of conference papers organized by Japanese academic societies. In preparing the data for the experiments, first 52,867 papers are from five different societies: 23,105 from the Society of Polymer Science, Japan (SPSJ), 20,482 from the Japan Society of Civil Engineers (JSCE), 4,832 from the Japan Society for Precision Engineering (JSPE), 2,434 from the Ecological Society of Japan (ESJ), and 2,014 from the Japanese Society for Artificial Intelligence (JSAI).The papers were then analyzed by the morphological analyzer ChaSen Ver.2.02 (Matsumotoetal., 1999) to extract nouns and compound nouns using the Part-Of-Speech tags. Next, the co-occurrence frequencies between documents and terms were collected. After preprocessing, the number of distinctive terms was 772,852 for the 52,867 documents.

2.3.1.3 CLUSTER EVALUATION

In the first experiments, a framework of unsupervised text categorization, where the quality of the generated clusters was evaluated by the goodness of the separation between different societies. Table 2.3 compares the total number of generated clusters (*c*), the average number of documents per cluster (*sd*), and the average number of terms per cluster (*st*), for different values of *d*. We also examined the ratio of unique clusters that consist only of documents from a single society (*r*s), and an inside-cluster ratio that is defined as the average relative weight of the dominant society for each cluster (*ri*). Here, the weight of each society within a cluster was calculated as the sum of the significance weights of its component documents given by Eq. The results shown in Table 1 indicate that reducing the value of *d* improves the quality of the generated clusters: with smaller *d*, the single society ratio and the inside-cluster ratio becomes higher, while the number of generated clusters becomes smaller.

δ	c	s_d	s_t	r_s	r_i
0.10	136,832	3.25	9.3	0.953	0.983
0.30	187,079	3.94	29.4	0.896	0.960
0.50	196,208	4.81	39.7	0.866	0.951
0.70	196,911	5.39	44.4	0.851	0.948
0.90	197,164	5.81	46.3	0.841	0.945
0.95	197,193	5.89	46.6	0.839	0.944

Table 2.3 Cluster evaluation

2.3.1.4 CATEGORIZATION RESULTS

In the second experiment, a framework of supervised text categorization, where the generated clusters were used as indices for classifying documents between the existing societies, and the categorization performance was examined.For comparison, two supervised text categorization methods, naive Bayes and Support Vector Machine (SVM), were also applied to the same training and test sets.Slightly better than naive Bayes, but not as good as SVM. We also compared the performance for varied sizes of training sets and also using different combination of societies, but the tendency remained the same.

δ	correct	judge	F-value
0.10	2,370	2,446	0.932
0.30	2,520	2,623	0.957
0.50	2,575	2,641	0.975
0.70	2,583	2,641	0.978
0.90	2,584	2,641	0.978
0.95	2,583	2,641	0.978
naive Bayes	2,579	2,641	0.977
SVM	2,602	2,641	0.985

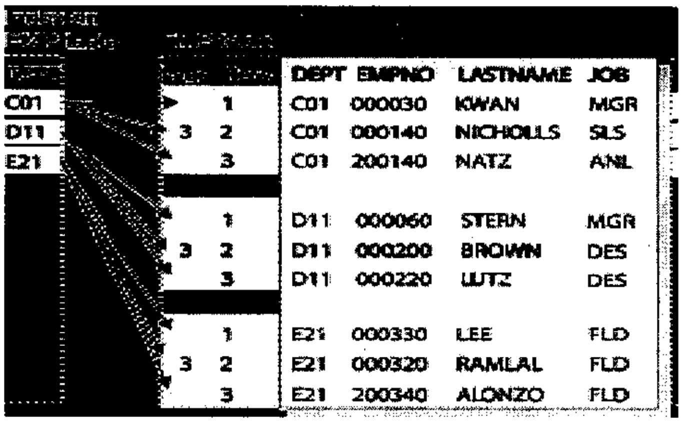
Table 2.4 Categorization results

For comparison purposes, in this only the conventional documents-and- terms feature space in the experiments are used. However, the proposed micro-clustering framework can be applied more flexibly to other cases as well. For example, generatation clusters using the co-occurrences of the triple of documents, terms, and authors are used .

Although the performance was not much different in terms of text categorization (2,584 correct judgments out of 2,639 judgments, the precision slightly improved), we can confirm that many of the highly ranked clusters contain documents produced by the same group of authors, emphasizing the characteristics of such generated clusters.

2.3.1.5 CLUSTER INDEXING

A clustered index determines the storage order of data in a table. Data records are ordered according to field that does not have distinct value for each record.
The field is called as the clustering field and the cluster index built on it. Cluster index key is used to order records in the table.
Dense Index: Data-file is ordered by the search key and every search key value has a separate index record. This structure requires only a single seek to find the first occurrence of a set of contiguous records with the desired search value.
Sparse Index: Data-file is ordered by the index search key and only some of the search key values have corresponding index records. Each index record's data-file pointer points to the first data-file record with the search key value.
A clustered index is analogous to a telephone directory, which arranges data by last name. Because the clustered index dictates the physical storage order of the data in the table, a table can contain only one clustered index. However, the index can comprise multiple columns (a composite index), like the way a telephone directory is organized by last name and first name.



	DEPT	EMPNO	LASTNAME	JOB
1	C01	000030	KWAN	MGR
2	C01	000140	NICHOLLS	SLS
3	C01	200140	NATZ	ANL
1	D11	000060	STERN	MGR
2	D11	000200	BROWN	DES
3	D11	000220	LUTZ	DES
1	E21	000330	LEE	FLD
2	E21	000320	RAMLAL	FLD
3	E21	200340	ALONZO	FLD

Figure 2.4 Cluster index

2.3.1.6 ADVANTAGES OF CLUSTER INDEXING:

Here a method of generating and overlapping micro-clusters in which documents, terms, and other related elements of text-based information are grouped together.
Comparing the proposed micro-clustering method with existing text categorization methods, the distinctive feature of the former is that the documents on borders are readily viewed and examined. In addition, the terms in the cluster can be further utilized in digesting the descriptions of the clustered documents. Such properties of micro-clustering may be particularly important when the system actually interacts with its users.

2.3.1.7 DISADVANTAGES OF CLUSTER INDEXING:

(i) Enhancing the probabilistic models considering other discounting techniques in linguistic studies; (ii) developing a strategy for initiating clusters by combining different attribute sets, such as documents or authors; and also (iii) establishing a method of evaluating overlapping clusters. Possibility of applying the proposed framework to Web document clustering problems are not possible with this method. [2]

2.4 HASH-BASED INDEXES

2.4.1 DESCRIPTION

Hash table is the data structure that uses hash function to map identifying values, known as keys to their associated values. The hash function is to transform the key index of (the hash) to the array element(bucket) where corresponding value is to be sought.
The hash function should map each possible key to a unique index, but it is rarely achievable in practice. Hash collision the situation where different keys happen to have the same hash value. The hash table the average cost for each look up is independent of the number of elements stored in the table. Many hash table designs allow arbitary insertions and deletions of key-value pairs

2.4.1.1 CHOOSING A GOOD HASH FUNCTION

Hash table algorithms calculate an index from the data items key and use this index to place the data into query. The implementation of this calculation is the hash function f

Index = f(key, array length).

The hash function calculates an index within the array from the Key. Array length in the size of array.A good Hash function and its implementation algorithm are essential for good hash table performance, but may be difficult to achieve. A basic requirement is that the function should provide a uniform distribution of hash values. A non-uniform distribution increases the number of collisions, and the cost of resolving them. It is implemented in statistical tests. The distribution needs to be uniform only for tables that occurs in the application. For open addressing schemas the hash function should also avoid clustering, the mapping of two or more keys to consecutive slots.

2.4.1.2 PERFECT HASH FUNCTION

If all keys are known ahead of time, a perfect hash function can be used to create a perfect hash table that has no collisions. If minimal perfect hashing is used, every location in hash table can be used. Perfect hashing allow for constant time lookups in the worst case.

USES:

Associate arrays

Database indexing

Caches

Sets

Object and unique data representation.

2.4.1.3 ADVANTAGES OF HASH FUNCTION:

The main advantage of hash table is data structures is speed. This advantages is more important when the number of entries is large. Hash tables are particularly efficient When the maximum number of entries can be predicted in advance, so that the bucket array can be allocated once with optimum size and never resized.

2.4.1.4 DISADVANTAGES OF HASH FUNCTION:

Hash tables are not more efficient if numbers of entries are small. Hash tables in general are poor for locality of references. [12]

2.5 BITMAP INDEX:

A bitmap index is a special kind of database index that uses bitmaps. A Bitmap index is a special type of structure used by DBMS to

- Optimize search
- Retrieval for low variability data
- Reduce response time for large ad-hoc queries.
- Dramatic performance gains on hardware with a relatively small amount of memory. Bitmap indexes are the most effective for the queries that include multiple conditions in where clause.Bitmap indexes are good for low cardinality columns have a small number of distinct values.

IDENTIFIER	GENDER	BITMAPS	
		F	M
1	FEMALE	1	0
2	MALE	0	1
3	MALE	0	1
4	UNSPECIFIED	0	0
5	FEMALE	1	0

Table 2.5 Gender-Female Vs Male

Bitmap indexes have traditionally been considered to work well for data such as gender, which has a small number of distinct values, for example male and female, but many occurrences of those values. This would happen if, for example, you had gender data for each resident in a city. Bitmap indexes have a significant space and performance advantage over other structures for such data. However, some researchers argue that Bitmap indexes are also useful for unique valued data which is not updated frequently. Bitmap indexes use bit arrays (commonly called bitmaps) and answer queries by performing bitwise logical operations on these bitmaps.Bitmap indexes are also useful in the data warehousing applications for joining a large fact table to smaller dimension table such as those arranged in a star schema.

Example: Continuing the gender example, a bitmap index may be logically viewed as follows: on the left, identifier refers to the unique number assigned to each customer, gender is the data to be indexed, the content of the bitmap index is shown as two columns under the heading bitmaps. Each column in the above illustration is a bitmap in the bitmap index. In this case, there are two such bitmaps, one for gender Female and one for gender Male. It is easy to see that each bit in bitmap *M* shows whether a particular row refers to a male. This is the simplest form of bitmap index. Most columns will have more distinct values. For example, the sales amount is likely to have a much larger number of distinct values. Variations on the bitmap index can effectively index this data as well. We briefly review three such variations. Bitmap indexes are used by DBMSs to accelerate decision support queries. The main advantage of Bitmap indexes is that complex logical selection operations can be performed very quickly by applying low-cost Boolean operations such as OR, AND, and NOT, thus, reducing search space before going to the primary source data.

2.5.1 DISADVANTAGES OF BITMAPINDEX:

- It works slow on high cardinality data [3].

CHAPTER 3
METHODOLOGY

3.1 DRAWBACKS OF EXISTING SYSTEM:

The bitmap index works slow on high cardinality data. To further speed up the query processing using bitmap index,the enhancement of bitmap index is made.

3.2 PROPOSED WORK:

In proposed work in order to speed up the query processing using bitmap index method enhancement is performed by using the following bitmap techniques such as simple bitmap techniques, range based techniques, bit sliced indexing techniques, projection indexing techniques.

3.2.1 PROJECTION INDEX:

A Projection Index on an indexed column name in a table Employee stores all values of name in the same order as they appear in another table. Each row of the Projection Index stores one value of name. The row order of value name in the index is the same as the row order of value name in Employee.

3.2.1.1 ADVANTAGES OF PROJECTION INDEX:

In general, the queries from a data warehouse retrieve only a few of the table's columns therefore, having the Projection Index on these columns reduces extremely the cost of querying because a single I/O operation may bring more values into memory. It speeds up the performance when a few columns in the table are retrieved

3.2.2 SIMPLE BITMAP INDEX:

- The Simple Bitmap Index consists of a collection of bitmap vectors each of which is created to represent each distinct value of the indexed column.
- The bit in a bitmap vector, representing value name, is set to 1 if the ith record in the indexed table contains the same name.

3.2.2.1 ADVANTAGES OF SIMPLE BITMAP INDEX:

- It is well suited for low cardinality columns.
- It utilizes bitwise operations.
- It uses low space.

3.2.2.2 RANGEBASED INDEX:

- The space complexity of the Simple Bitmap index is low for low cardinality attributes but large for high cardinality attributes.
- The most important idea of Range-Based Indexes is to reduce storage overhead by partitioning, That is, attribute values are split into smaller number of ranges and represented by bitmap vectors. Indeed, a bit is set to 1 if a record falls into specified range otherwise this bit is set to 0.

3.2.2.3 ADVANTAGE OF RANGEBASED INDEX:

- The great advantage of a Range-Based index over the Simple Bitmap index is that only a lower number of bitmap vectors need to be stored.

3.2.2.4 BITSLICED INDEX:

- Bit-Sliced Index is a set of bitmap slices which are orthogonal to the data held in a projection index.
- A bit-sliced index based on converting integer values to binary values in order to perform fast logical operations on them since that hardware support directly.
- choose an optimal number of bits per bit-vector in order to represent the whole attribute domain and to occupy minimum space.[4]

3.2 SYSTEM MODULE:

MODULES:

- **MODULE I:** Implementing three indexing techniques such as Bitmap,Cluster,Hash and their performance comparision with different datawarehouse queries and with respect to time.
- **MODULE II**: Enhancement of Bitmap indexing techniques such as Simple,Projection,RangeBased,Bitsliced and Encoded Index and comparing the performance of above techniques with different datawarehouse queries and with respect to time.

3.3.1 BRIEF DESCRIPTION:

MODULE I:

- Creating the database.
- Implementing three indexing techniques such as Bitmap,Cluster,Hash indexes and connecting a large database with the techniques.
- Comparing the performance of three indexing techniques with different datawarehouse queries with respect to time and plotting it in graph.

MODULE II:

- Enhancement of Bitmap indexing Such as Simple Bitmap, RangeBased, Projection, Encoded and Bit Sliced Index.

CHAPTER 4:

4.1 SYSTEM SPECIFICATION:

4.1.1 Hardware Requirement

Processor	:	Pentium IV
Speed	:	Above 500 MHz
RAM capacity	:	2 GB
Hard disk drive	:	80 GB
Key Board	:	Samsung 108 keys
Mouse	:	Logitech Optical Mouse
Printer	:	DeskJet HP
Motherboard	:	Intel
Cabinet	:	ATX
Monitor	:	17" Samsung

4.1.2 Software Requirement

Operating System	:	Windows XP and above
Back End	:	MYSQL Server

4.2 TECHNOLOGY INFRASTRUCTURE

4.2.1 JAVA:

The Java technology is:

- A programming language
- A development environment
- An application environment
- A deployment environment

4.2.1.1JAVA TECHNOLOGY:

An Application and Runtime Environment

Java technology applications are typically general-purpose programs that run on any machine where the Java runtime environment (JRE) is installed.

There are two main deployment environments:

- The JRE supplied by the Java 2 Software Development Kit (SDK) contains the complete set of class files for all the Java technology
- packages, which includes basic language classes, GUI component classes, and so on.
- The other main deployment environment is on your web browser.
- Most commercial browsers supply a Java technology interpreter and
- runtime environment.

4.2.1.2 Java Features

Some features of Java:

The Java Virtual Machine

Garbage Collection

Code Security

Code Security

Code security is attained in Java through the implementation of its Java Runtime Environment (JRE).

JRE

Runs code compiled for a JVM and performs class loading (through the class loader), code verification (through the bytecode verifier) and finally code execution

Phases of a Java Program

The following figure describes the process of compiling and executing a Java program

Java Background

A programming language, development environment, application environment

and deployment environment

Java Features

Java Virtual machine, garbage collection and code security

Phases of a Java Program

Write, compile, run

4.2.1.3 JAVA IS SIMPLE:

Java has automatic memory management

No dangling pointers.

No memory leaks.

Java simplifies pointer handling

No reference/deference operations

No make files/No header files

C++ syntax streamlined.

4.2.1.4 JAVA IS OBJECT ORIENTED

All functions are associated with objects:

Some describe it object- obssesed

Almost all datatypes

Files, arrays, strings, sockets.

Object is a common ancestor of all classes.

JAVA VERSIONS:

Java 1.0 released in 1995

Java 1.1 released in early 1997

A new event-handling model based on listeners.

Remote method invocation(RMI) and object sterilization.

JavaBeans component architecture

Digitally signed applets to extended security privileges without resorting to all or nothing model of browser plug-ins or ActiveX.

JAVA VERSION SUPPORTS:

APPLETS:

Use JDK1.1

Internet Explorer 4.0

APPLICATIONS:

JDK1.4

BEST APPROACH:

Use JDK1.4, but bookmark the JDK1.1 API to check methods writing applets.

4.2.1.5 JAVA FEATURES:

- Well defined primitive data types: int, float, double char.
- Interfaces
- Exceptions
- Concurency

4.2.1.6 THE JAVA PROGRAMMING ENVIRONMENT:

Java programming language specification.

Syntax of java programs.

Defines different constructs and their semantics.

JAVA BYTE CODE:

Intermediate representation for java programs.

Java compiler: Transform java programs into java bytes.

Java interpreter: Read programs written in java byte code and execute them

4.2.1.7 JAVA PROGRAMMING ENVIRONMENT:

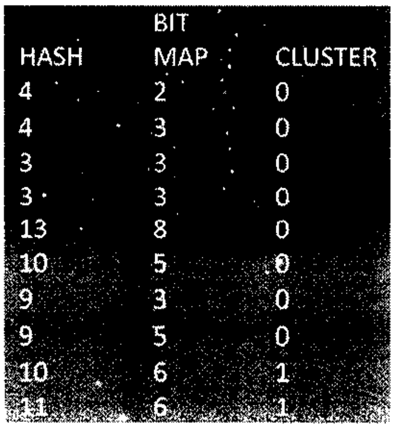
Set of libraries that provide services such as GUI, data structures, etc.

CHAPTER 5

5.1 EXPERIMENTAL RESULTS AND DISCUSSION:

Here the comparision of three indexing techniques such as bitmap indexing techniques, cluster indexing techniques and Hash-Based indexing techniques are compared with different size of data from employee database which contains 900 records. To retrieve the data from the particular employee databases we are executing the query. The particular query execution time is calculated. Here the hash index takes a large query execution time than other indexing techniques. Because it takes large time to retrieve the records from the database.

5.1.1 COMPARISION OF INDEXING TECHNIQUES WITH DIFFERENT SIZE OF DATA:



HASH	BIT MAP	CLUSTER
4	2	0
4	3	0
3	3	0
3	3	0
13	8	0
10	5	0
9	3	0
9	5	0
10	6	1
11	6	1

Table 5.6 Comparision of Indexing techniques with different size of data

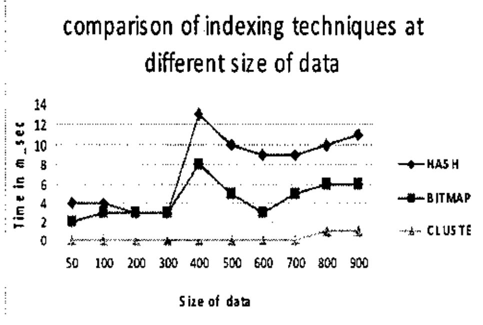
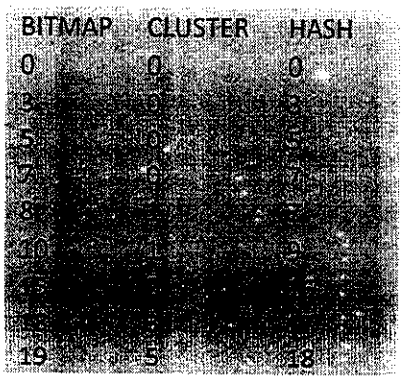


Figure 5.5 Comparision of Indexing techniques with different size of data

5.1.2 ANALYSIS OF FULL TABLE SCAN QUERY ON ALL TECHNIQUES

Above results illustrate that full table scan query performs well on cluster indexes whereas other two techniques takes significant time as compare to cluster index. The reason for this is possibly the implementation of these techniques and their storage in the database, as we stored them in the database in order to execute all queries one by one. Another reason for this is that each retrieved result need to search two tables
for each retrieval record. One search from index table and other is the original data table



BITMAP	CLUSTER	HASH
0	0	0
3	0	3
5	0	5
7	0	7
8	[illegible]	[illegible]
10	1	9
[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]
19	5	18

Table 5.7 Analysis of full table scan query on all techniques

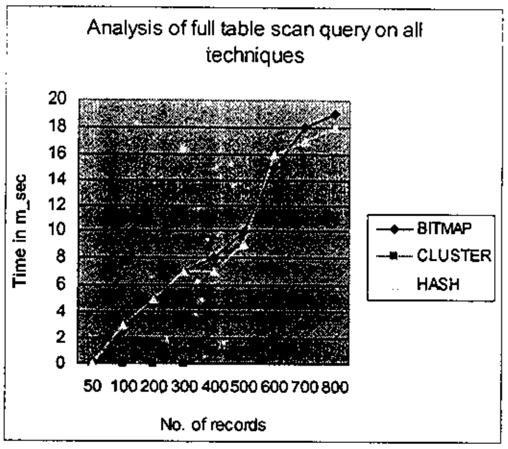
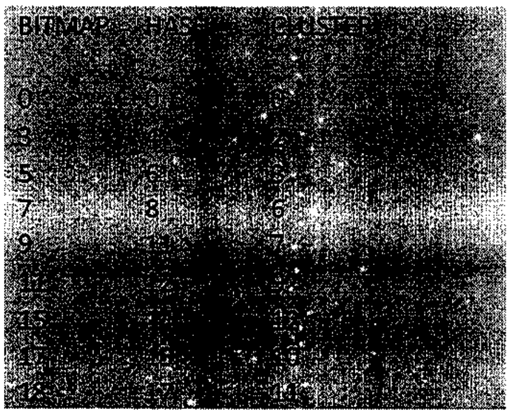


Figure 5.6 Analysis of full table scan query on all techniques

5.1.3 ANALYSIS OF EXACT MATCH QUERY ON ALL TECHNIQUES

Exact match query takes almost same time on all techniques at small data sets but as data size increases hash based indexing shows good results.



BITMAP	HASH	CLUSTER
0	0	0
3	[illegible]	[illegible]
5	6	[illegible]
7	8	6
9	[illegible]	7
[illegible]	[illegible]	[illegible]
15	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]

Table 5.8 Analysis of exact match query on all techniques

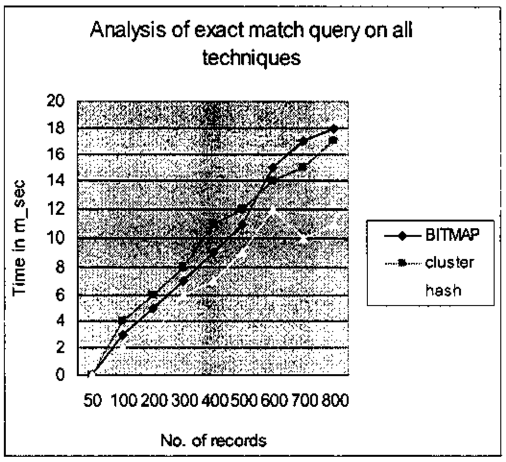
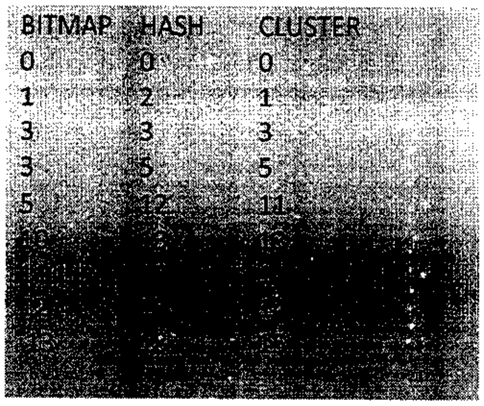


Figure 5.7 Analysis of exact match query on all techniques

5.1.4 ANALYSIS OF STATISTICAL (COUNT) QUERY ON ALL TECHNIQUES:

Statistical query such as Count() executes very well on bitmap index even when the data sets are large and gives result in less time as compared to others Bitmap index perform well because it only need to count I's in the queried bitmap without accessing original data table. Whereas hash index need to generate hash code and than to count the whole chain by parsing it so it take large time. Cluster index access original table so it results in longer time.



BITMAP	HASH	CLUSTER
0	0	0
1	2	1
3	3	3
3	5	5
5	12	11
10	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]

Table5.9 Analysis of statistical (count) query on all techniques

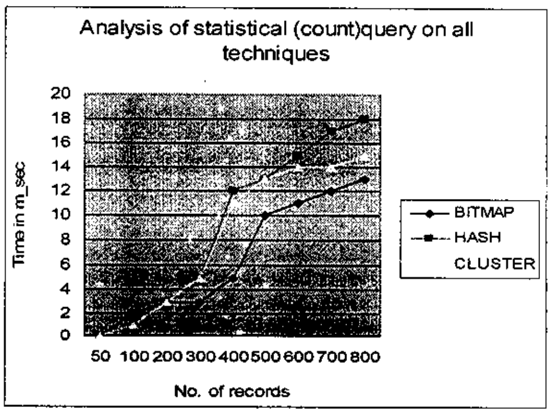
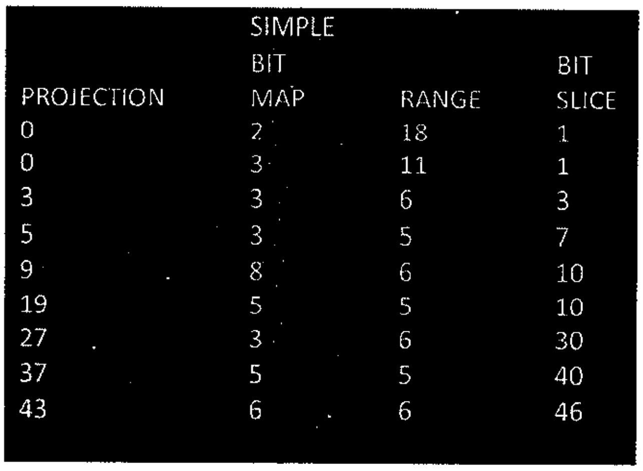


Figure5.8 Analysis of statistical (count) query on all techniques

5.1.5 COMPARISION OF INDEXING TECHNIQUES WITH DIFFERENT SIZE OF DATA:

Here the comparision of Simple bitmap index, Range Based index, Projection index,



PROJECTION	SIMPLE BIT MAP	RANGE	BIT SLICE
0	2	18	1
0	3	11	1
3	3	6	3
5	3	5	7
9	8	6	10
19	5	5	10
27	3	6	30
37	5	5	40
43	6	6	46

Table 5.10 Comparision of Indexing techniques with different size of data

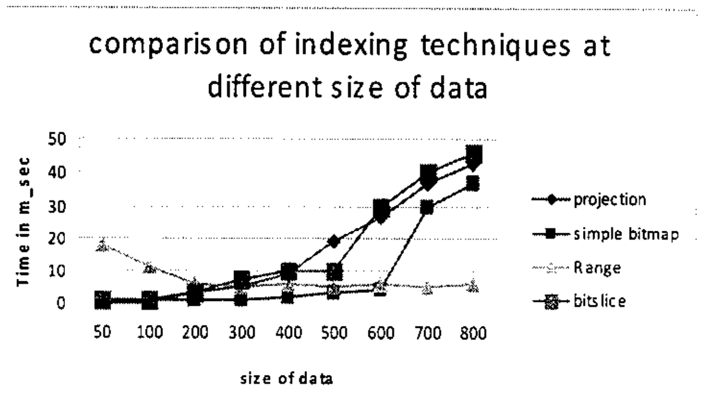


Figure 5.9 Comparision of Indexing techniques with different size of data

Bit-Sliced index are made and the Bit-sliced index shows large execution time than other indexing techniques, and the graphs are plotted according to the time Vs data.

CHAPTER 6

6.1 CONCLUSION AND FUTURE WORK:

Here the comparision of four indexing techniques such as bitmap indexing techniques, cluster indexing techniques and Hash-Based indexing techniques are compared with different size of data from employee database which contains 900 records. To retrieve the data from the particular employee databases we are executing the query. The particular query execution time is calculated. Here the Bit-slice index takes a large query execution time than other indexing techniques. Because it takes large time to convert the integer values to binary values. In the future work the Analysis of four indexing techniques with different datawarehouse queries are to be made and to select the best indexing techniques among the four indexing techniques.

CHAPTER 7

APPENDICES

SOURCE CODE:

Bitmap index:

```
import java.io.*;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.*;
import java.lang.*;
import java.util.*;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;
import java.util.Set;
class bitmap1
{
public static void main (String args[])
 {
 DataInputStream in = new DataInputStream (System.in);
 String prod[] = new String[100];
 String prod1[] = new String[100];
 String temp[] = new String[100];
 String temp1[] = new String[100];
 int[][]bitmap1 = new int[100][100];
   int list[][] = new int[100][100];
  int h;
   int[][]bitmap = new int[100][100];
   int i,j,l,n,k,g,d,f,n1,mark=0;
   Connection con = null;
```

```
Statement stmt,stmt1;
ResultSet rs,rs1,rs2,rs3,rs4;
HashMap map=new HashMap();
HashMap map1=new HashMap();
 Try
  {
 Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
 con=DriverManager.getConnection ("jdbc:odbc:Employee","root","root");

stmt=con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,ResultSet.CONCU
R_READ_ONLY);
rs = stmt.executeQuery("SELECT PRODUCT FROM 100record");

stmt1=con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,ResultSet.CONC
UR_READ_ONLY);

 i=0;
 j =0;
 while(rs.next())
 {
 prod[i] = rs.getString("PRODUCT");
System.out.println((prod[i]) + " ");
 i++;
  }
n=i-1;
rs1 = stmt.executeQuery("SELECT distinct PRODUCT FROM 100record");
 i=0;
 while(rs1.next())
 {
 temp[i] = rs1.getString("PRODUCT");
 System.out.println((temp[i]) + " ");
```

```
  i++;
   }
 int distinctvalues=i-1;
if(!con.isClosed())
System.out.println("Successfully connected to " +"MySQL server using TCP/IP...");
 for(j=0;j<=n;j++)
 {
 for(l=0;l<=distinctvalues;l++)
 {
if(temp[l].equals(prod[j]))
{
 bitmap[j][l]=1;
  }
else
{
bitmap[j][l]=0;
  }
}
}
for(j=0;j<=n;j++)
{
for(l=0;l<=distinctvalues;l++)
{
System.out.print(bitmap[j][l]+"\t");
 map.put(list[j][l],bitmap[j][l]);
}
System.out.println();
  }
 g=0;
d=0;
 rs2 = stmt1.executeQuery("SELECT NAME FROM 100record");
```

```
while(rs2.next())
  {
prod1[g] = rs2.getString("NAME");
System.out.println((prod1[g]) + " ");
 g++;
 }
 n1=g-1;
 rs3=stmt1.executeQuery("SELECT distinct NAME  FROM  100record");
 g=0;
while(rs3.next())
{
temp1[g] = rs3.getString("NAME");
System.out.println((temp1[g]) + " ");
g++;
}
int distinct=g-1;
if(!con.isClosed())
System.out.println("Successfully connected to " +"MySQL server using TCP/IP...");
Calendar cal = Calendar.getInstance();
int min = cal.get(Calendar.MINUTE);
int tim = cal.get(Calendar.SECOND);
int time1 = min * 60 + tim;
for(g=0;g<=n;g++)
{
for(d=0;d<=distinct;d++)
{
System.out.print(bitmap1[g][d]+"\t");
map.put(list[g][d],bitmap[g][d]);
 }
System.out.println();
 }
 Set set = map.keySet();
Iterator iter = set.iterator();
 while (iter.hasNext())
  {
 Integer m = (Integer) iter.next();
 int s = (Integer) map.get(m);
 System.out.print(s);
  }
  i=47;
 System.out.println();
 for(h=0;h<=distinctvalues;h++)
 {
     System.out.print(bitmap[i][h]);
         }
     System.out.print ("The name is");
     System.out.println();
     g=3;
    System.out.println();
    for(f=0;f<=distinct;f++)
    {
     System.out.print(bitmap[g][f]);
    }
   cal = Calendar.getInstance();
   min = cal.get(Calendar.MINUTE);
   int tmr = cal.get(Calendar.SECOND);
   int time2 = min * 60 + tmr;
    int time=time2-time1;
   System.out.println("time");
   System.out.print(time + " ");
    }
 catch(Exception e)
 {
  System.err.println("Exception: " + e.getMessage());
  }
 finally
{
 try
   {
 if(con != null)
 con.close();
  }
 catch(Exception e)
   {
 System.out.println(e);
  }
 }
}
```

CLUSTER INDEX:

```
import java.io.*;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.*;
import java.lang.*;
import java.util.*;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;
import java.util.Set;
class Clusterin
```

```
 {
public static void main (String[] args)
String name[] = new String[50];
 String prod[] = new String[50];
String listArray[] = new String[50];
String listArray1[] = new String[50];
int length = args.length;
Connection con = null;
Statement stmt,stmt1,stmt2,stmt3;
ResultSet rs;
ResultSet rs1,rs2,rs3;
HashMap map=new HashMap();
HashMap map1=new HashMap();
try
{
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
con = DriverManager.getConnection("jdbc:odbc:Employee","root","root");

stmt=con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,ResultSet.CONCUR
_READ_ONLY);

stmt1=con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,ResultSet.CONCU
R_READ_ONLY);

stmt2=con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,ResultSet.CONCU
R_READ_ONLY);
rs = stmt.executeQuery("SELECT PRODUCT FROM 50record ");
int i=0;
while (rs.next())
{
name[i] = rs.getString("PRODUCT");
System.out.println((name[i]) + " ");
 i++;
 }
if(!con.isClosed())
 System.out.println("Successfully connected to " +"MySQL server using Tcp/Ip..");
 String temp = null;
 for( i = 0;i<50 ;i++)
 {
  for(int j = i+1; j<50;j++)
 {
 if(name[j].compareTo(name[i]) < 0)
 {
temp = name[i];
name[i] = name[j];
name[j] = temp;
  }
  }
  }
Calendar cal = Calendar.getInstance();
 int min = cal.get(Calendar.MINUTE);
 int tim = cal.get(Calendar.SECOND);
 int time1 = min * 60 + tim;
 System.out.print("The map is");
  for( i= 0;i < 50; i++)
  {
  System.out.println(name[i]);
  map.put(listArray[i],name[i]);
   }
  Set set = map.keySet();
 Iterator iter = set.iterator();
 while (iter.hasNext())
```

```
{
String o= (String)iter.next();
String p= (String)map.get(o);
System.out.print(p);
 }
String temp1 = null;
int k=0;
rs1 = stmt1.executeQuery("SELECT NAME From 100record");
System.out.println("The name is");
 while (rs1.next())
 {
 prod[k]=rs1.getString("NAME");
 System.out.println((prod[k])+ " ");
 k++;
 }
 if(!con.isClosed()
 System.out.println("Successfully connected to " +"MySQL server using Tcp/Ip..");
  for( k = 0;k<50;k++)
  {
  for(int j = k+1; j<50;j++)
  {
  if(prod[j].compareTo(prod[k]) < 0)
  {
  temp1 = prod[k];
  prod[k] = prod[j];
  prod[j] = temp1;
   }
   }
  }
 for( k= 0;k <50; k++)
{
```

```
System.out.println(prod[k]);
map1.put(listArray1[k],prod[k]);
}
while (iter.hasNext())
{
 String m = (String)iter.next();
String s = (String)map1.get(
System.out.print(s);
 }


k=0;
rs3=stmt2.executeQuery("SELECT        NAME        FROM        50record        where
PRODUCT='Enron'");
 System.out.println("The name is");
 while (rs3.next())
          {
   prod[k]=rs3.getString("NAME");
   System.out.println((prod[k])+ " ");
    k++;
     }
   cal = Calendar.getInstance();
   min = cal.get(Calendar.MINUTE);
   int tmr = cal.get(Calendar.SECOND);
   int time2 = min * 60 + tmr;
   int time=time2-time1;
   System.out.println("time");
    System.out.print(time + " ");
     }
    catch (Exception e)
    {
    System.err.println ("Exception: " + e.getMessage());
```

```
}
finally
{
try
{
if (con != null)
con.close ();
 }
catch (Exception e)
{
System.out.println(e);
 }
 }
 }
 }
```

HASH BASED INDEX:

```
import java.io.*;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.*;
import java.util.*;
import java.util.Arrays;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;
import java.util.Set;
import static java.lang.System.out;
class Adr2
```

```
{
 public static void main(String args[])
  {
 Connection con = null;
 Statement stmt,stmt1;
 ResultSet rs,rs1;
 HashMap map=new HashMap();
 DataInputStream in = new DataInputStream(System.in);
 int number;
 int[] anArray;
 anArray = new int[100];
 int[] pin;
 pin = new int[100];
 int temp,k;
  try
  {
 Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
 con =DriverManager.getConnection("jdbc:odbc:Employee","root","root");

stmt=con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,ResultSet.CONCUR
_READ_ONLY);

stmt1=con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,ResultSet.CONCU
R_READ_ONLY);
 rs = stmt.executeQuery("SELECT RECORD FROM 100record");
 int i=0;
while(rs.next())
  {
anArray[i] = rs.getInt("RECORD");
System.out.println((anArray[i]) + " ");
 i++;
```

```
  }
 if(!con.isClosed())
 System.out.println("Successfully connected to " +"MySQL server using TCP/IP...");
System.out.println("Enter the number");
 int num = Integer.parseInt(in.readLine());
 System.out.println("num = " + num);
 for(i=0;i<100;i++)
  {
  temp = (anArray[i] % num);
  switch(temp)
   {
 case 0:
 System.out.println("The values in count[0] is" + anArray[i]);
 break;
  case 1:
  System.out.println("The values in count[1] is" + anArray[i] );
  break;
  case 2:
  System.out.println("The value in count[2] is" + anArray[i]);
  break;
  case 3:
  System.out.println("The value in count[3] is" +  anArray[i]);
  break;
  case 4:
  System.out.println("The value in count[4] is" + anArray[i]);
  break;
  case 5:
  System.out.println("The value in count[5] is" + anArray[i]);
  break;
  case 6:
  System.out.println("The value in count[6] is" + anArray[i]);
break;
case 7:
System.out.println("The value in count[7] is" + anArray[i]);
 break;
case 8:
System.out.println("The value in count[8] is" + anArray[i]);
break;
 case 9:
  System.out.println("The value in count[8] is" + anArray[i]);
  break;
   }
  }
   Calendar cal = Calendar.getInstance();
   int min = cal.get(Calendar.MINUTE);
    int tim = cal.get(Calendar.SECOND);
    int time1 = min * 60 + tim;
    System.out.println("The record");
    for(i=0;i<100;i++)
   {
     k=0;
     System.out.println(anArray[i]);
     map.put(pin[k],anArray[i]);
     System.out.print(pin[k]);
     k++;
     }
    Set set = map.keySet();
     Iterator iter = set.iterator();
     while (iter.hasNext())
     {
     Integer m = (Integer)iter.next();
     int s = (Integer)map.get(m);
 System.out.print(s);
  }
 System.out.println("The value is");
 rs1=stmt1.executeQuery("SELECT RECORD FROM 100record where ID=5");
 i=0;
while(rs1.next())
 {
 anArray[i] = rs1.getInt("RECORD");
 System.out.println((anArray[i]) + " ");
  i++;
  }
  cal = Calendar.getInstance();
  int tmr = cal.get(Calendar.SECOND);
  int time2 = min * 60 + tmr;
  int time=time2-time1;
System.out.println(time + " ");
   }
catch(Exception e)
 {
 System.err.println ("Exception: " + e.getMessage());
  }
finally
 {
 try
  {
  if (con != null)
  con.close ();
   }
 catch(Exception e)
  {
   System.out.println(e);
```

```
    }
   }
   }
  }
```

Proposed work:

SIMPLE BITMAP INDEX:

```
import java.io.*;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.*;
import java.lang.*;
import java.util.*;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;
import java.util.Set;
class bitmap2
{
public static void main(String args[])
  {
DataInputStream in = new DataInputStream(System.in);
 String prod[] = new String[100];
 String prod1[] = new String[100];
 String temp[] = new String[100];
 String temp1[] = new String[100];
  int[][]bitmap1 = new int[100][100];
  int list[][] = new int[100][100];
  int h,temp2,temp3;
 int lac[][] = new int[100][100];
 int jac[][] = new int[100][100];
 int[][]bitmap = new int[100][100];
 int pin[][]=new int[100][100];
 int i,j,l,n,k,g,d,f,n1,mark=0;
 Connection con = null;
  Statement stmt,stmt1;
  ResultSet rs,rs1,rs2,rs3,rs4;
  HashMap map=new HashMap();
  HashMap map1=new HashMap();
  try
  {
cal.get(Calendar.MINUTE);
con=DriverManager.getConnection("jdbc:odbc:Employee","root","root");

stmt=con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,ResultSet.CONCUR
_READ_ONLY);
rs = stmt.executeQuery("SELECT PRODUCT FROM 100record");
stmt1=con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,ResultSet.CONCU
R_READ_ONLY);
i=0;
j=0;
while(rs.next()
 {
prod[i] = rs.getString("PRODUCT");
System.out.println((prod[i]) + " ");
 i++;
  }
 n=i-1;
 rs1 = stmt.executeQuery("SELECT distinct PRODUCT FROM 100record");
 i=0;
 while(rs1.next())
{
temp[i] = rs1.getString("PRODUCT");
System.out.println((temp[i]) + " ");
 i++;
  }
 int distinctvalues=i-1;
 if(!con.isClosed())
 System.out.println("Successfully connected to " +"MySQL server using TCP/IP...");
 for(j=0;j<=n;j++)
 {
for(l=0;l<=distinctvalues;l++)
 {
if(temp[l].equals(prod[j]))
 {
 bitmap[j][l]=1;
  }
   else
   {
   bitmap[j][l]=0;
   }
  }
  }
   for(j=0;j<=n;j++)
  {
   for(l=0;l<=distinctvalues;l++)
   {
    System.out.print(bitmap[j][l]+"\t");
     map.put(list[j][l],bitmap[j][l]);
      }
     System.out.println();
  }
 g=0;
 d=0;
rs2 = stmt1.executeQuery("SELECT NAME FROM 100record");
while(rs2.next())
 {
 prod1[g] = rs2.getString("NAME");
 System.out.println((prod1[g]) + " ");
  g++;
  }
 n1=g-1;
 rs3= stmt1.executeQuery("SELECT distinct NAME FROM 100record");
 g=0;
while(rs3.next())
{
temp1[g] = rs3.getString("NAME");
System.out.println((temp1[g]) + " ");
 g++;
}
 int distinct=g-1;
 if(!con.isClosed())
 System.out.println("Successfully connected to " +"MySQL server using TCP/IP...");

Calendar cal = Calendar.getInstance();
 int min = cal.get(Calendar.MINUTE);
 int tim = cal.get(Calendar.SECOND);
 int time1 = min * 60 + tim;
for(g=0;g<=n;g++)
{
for(d=0;d<=distinct;d++)
 {
   System.out.print(bitmap1[g][d]+"\t");
  }
 System.out.println();
 }
 Set set = map.keySet();
  Iterator iter = set.iterator;
 while (iter.hasNext())
  {
  Integer m = (Integer)iter.next();
  int s = (Integer)map.get(m);
  System.out.print(s);
   }
  h=0;
  map.put(pin[h][d],map);
 System.out.print(pin[h][d]+"\t");
  h++;
 d++;
  i=3;
  System.out.println();
  for(h=0;h<=distinctvalues;h++)
  {
  System.out.print(bitmap[i][h]);
   }
 System.out.print("The name is");
 System.out.println();
  g=3;
  System.out.println();
  for(f=0;f<=distinct;f++)
 {
  System.out.print(bitmap[g][f]);
   }
  System.out.println();
   h=3;
   f=3;
 System.out.println();
  for(h=0;h<=distinctvalues;h++)
  {
  System.out.print(bitmap[i][h]&bitmap[g][f]);
  }
  cal = Calendar.getInstance();
  min = cal.get(Calendar.MINUTE);
  int tmr = cal.get(Calendar.SECOND);
  int time2 = min * 60 + tmr;
  int time=time2-time1;
  System.out.println("time");
   System.out.print(time + " ");
 }
 catch(Exception e)
  {
 System.err.println("Exception: " + e.getMessage());
  }
finally
{
Try
{
if(con != null)
con.close();
}
catch(Exception e)
{
System.out.println(e);
 }
```

```
}
}
```

RANGE BASED INDEX:

```
import java.io.*;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.*;
import java.util.*;
import java.lang.*;
 class rangemat
  {
  public static void main(String args[])
   {
 Connection con = null;
 Statement stmt;
 ResultSet rs;
 DataInputStream in = new DataInputStream(System.in);
 int[] anArray;
 anArray = new int[100];
 int[]bit = new int[100];
 try
 {
 Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
 con = DriverManager.getConnection("jdbc:odbc:Employee","root","root");

stmt=con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,ResultSet.CONCUR
_READ_ONLY);
 rs = stmt.executeQuery("SELECT record FROM 100record");
 Calendar cal = Calendar.getInstance();
 int min = cal.get(Calendar.MINUTE);
int tim = cal.get(Calendar.SECOND);
int time1 = min * 60 + tim;
int i=0;
while(rs.next())
 {
anArray[i] = rs.getInt("record");
 System.out.println((anArray[i]) + " ");
 i++;
 }
 if(!con.isClosed())
 System.out.println("Successfully connected to " +"MySQL server using TCP/IP...");
 System.out.println("Enter the number");
 int num = cal.get(Calendar.MINUTE);
 System.out.println("num = " + num);
 for(i =0;i<100;i++)
  {
if(anArray[i]<num)
 {
  bit[i] = 1;
  }
 else
 {
  bit[i] = 0;
  for( i=0;i<100;i++)
 {
 System.out.print(" " + bit[i]);
  }
 cal = Calendar.getInstance();
 min = cal.get(Calendar.MINUTE);
 int tmr = cal.get(Calendar.SECOND);
int time2 = min * 60 + tmr;
int time=time2-time1;
System.out.println("time");
System.out.println(time + " ");
 }
catch(Exception e)
{
 System.out.println(e);
}
}
}
```

BITSLICED INDEX:

```
import java.io.*;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.*;
import java.lang.*;
import java.util.*;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;
import java.util.Set;
public class Bitslice2
{
 public static void main(String[] args)
{
DataInputStream in = new DataInputStream(System.in);
int[][] binaryNumber=new int[50][50];
int[][] bitslice=new int[50][50];
```

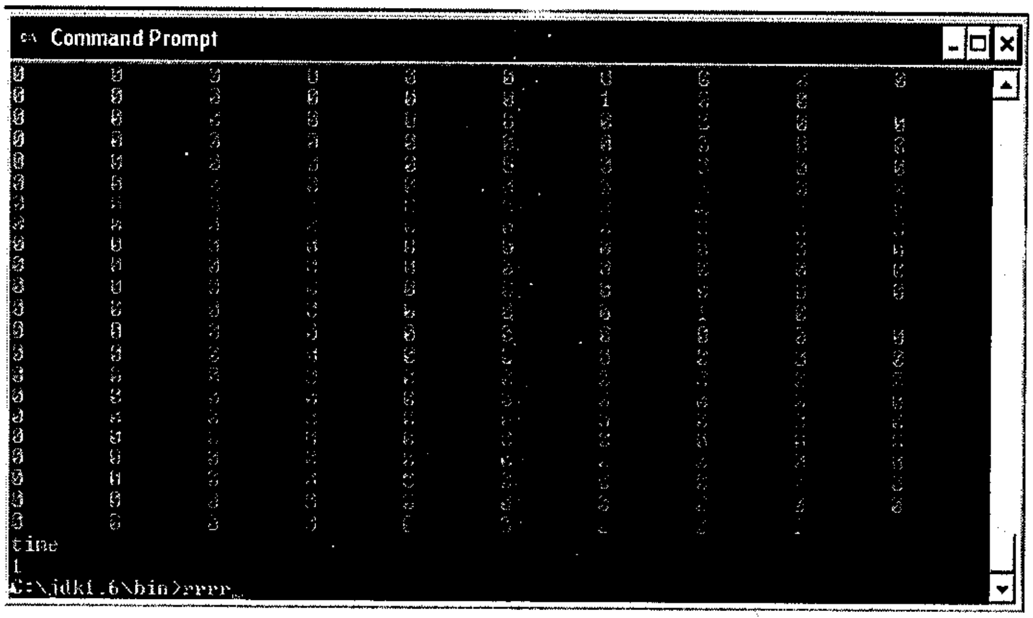
```
int[] decimalNumber= new int[50];
int[]temp=new int[50];
 int[] temp1=new int[50];
int max=0,i,j,k,l,c,temp2;
Connection con = null;
Statement stmt,stmt1,stmt2,stmt3;
ResultSet rs;
ResultSet rs1;
for(i=0;i<50;i++)
for(j=0;j<50;j++)
binaryNumber[i][j]=0;
HashMap map=new HashMap();
HashMap map1=new HashMap();
try
{
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
con=DriverManager.getConnection("jdbc:odbc:Employee","root","root");
stmt=con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,ResultSet.CONCUR
_READ_ONLY);
 rs = stmt.executeQuery("SELECT RECORD FROM 50record");
   i=0;
  Calendar cal = Calendar.getInstance();
  int min = cal.get(Calendar.MINUTE);
  int tim = cal.get(Calendar.SECOND);
  int time1 = min * 60 + tim;
  while (rs.next())
 {
decimalNumber[i] = rs.getInt("RECORD");
 System.out.println((decimalNumber[i]) + " ");
 i++;
}
```

```
 if(!con.isClosed())
 System.out.println("Successfully connected to " +"MySQL server using Tcp/Ip..");
for(i=1;i<50;i++)
  {
if (decimalNumber[i] <= 0)
System.out.println("ERROR: entered integer is nonpositive.");
else
 {
 j=1;
 while (decimalNumber[i] != 0)
 {
 temp[j]=(decimalNumber[i] % 2);
 decimalNumber[i] /= 2;
j++;
 }
l=1;
 System.out.println();
  for(k=j-1;k>=1;k--)
 {
binaryNumber[i][l]=temp[k];
System.out.print(binaryNumber[i][l]);
l++;
  }
 temp1[i]=j-1;
 if(i==1)
 max=temp1[i];
else
 {
 if(max<temp1[i])
max=temp1[i];
 }
 System.out.println();
  for(l=1;l<=j-1;l++)
  {
   System.out.println(binaryNumber[i][l]);
   }
   }
  System.out.print("");
  }
 System.out.println("HI "+max);
for(i=1;i<50;i++)
{
if(temp1[i]<max)
{
temp2=max-temp1[i];
for(j=1;j<=temp2;j++)
bitslice[i][j]=0;
k=1;
for(int t=temp2+1;t<=max;t++)
{
bitslice[i][t]=binaryNumber[i][k];
k=k+1;
}
temp1[i]=max;
}
else
{
for(k=1;k<=max;k++)
{
bitslice[i][k ]= binaryNumber[i][k];
}
}
```

```
}
 for(i=1;i<50;i++)
{
System.out.println();
for(j=1;j<50;j++)
{
System.out.print(bitslice[j][i]+"\t");
}
cal = Calendar.getInstance();
min = cal.get(Calendar.MINUTE);
int tmr = cal.get(Calendar.SECOND);
 int time2 = min * 60 + tmr;
 int time=time2-time1;

 System.out.println("time");
 System.out.println(time + " ");
  }
  catch (Exception e)
  {
 System.err.println ("Exception: " + e.getMessage());
  }
 finally
 {
 Try
{
if (con != null)
con.close ();
}
```

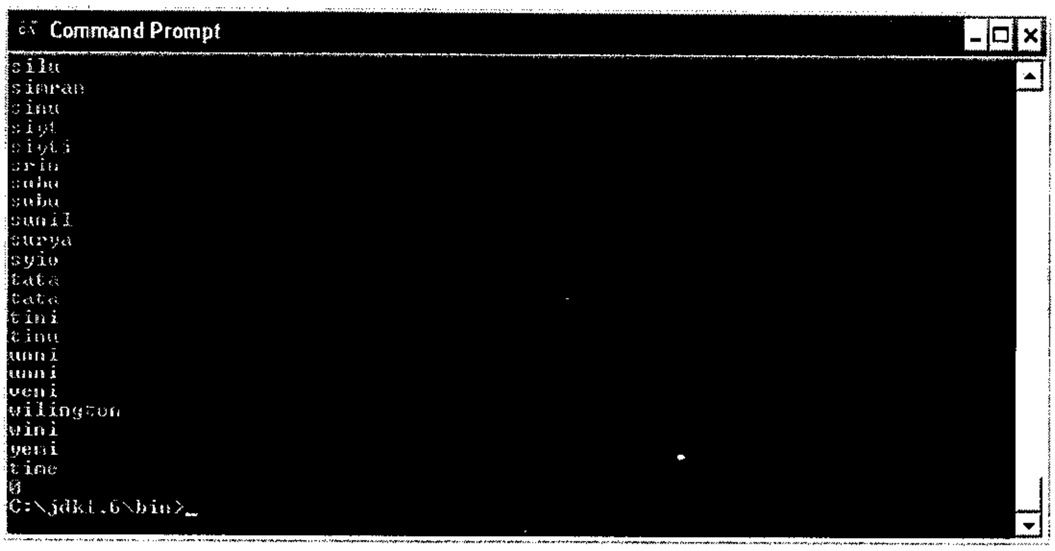
SNAPSHOTS:

BITMAP INDEX:



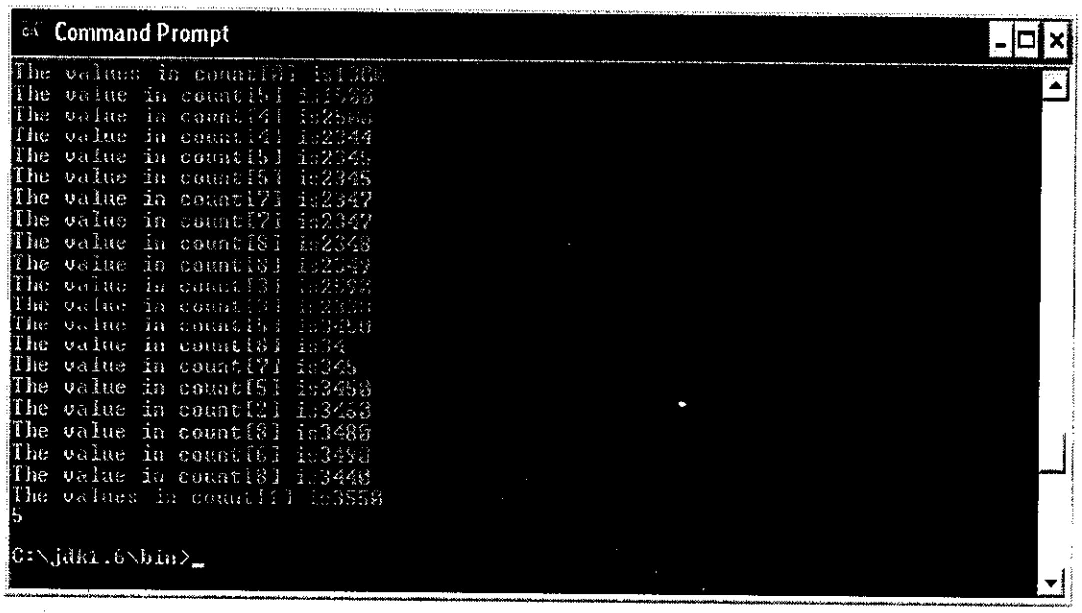
```
0       0       0       0       0       0       0       0       0       0
0       0       0       0       0       0       1       0       0
0       0       0       0       0       0       0       0       0       0
0       0       0       0       0       0       0       0       0       0
0       0       0       0       0       0       0       0       0       0
0       0       0       0       0       0       0       0       0       0
0       0       0       0       0       0       0       0       0       0
0       0       0       0       0       0       0       0       0       0
0       0       0       0       0       0       0       0       0       0
0       0       0       0       0       0       0       0       0       0
0       0       0       0       0       0       0       0       0       0
0       0       0       0       0       0       0       0       0
0       0       0       0       0       0       0       0       0       0
0       0       0       0       0       0       0       0       0       0
0       0       0       0       0       0       0       0       0       0
0       0       0       0       0       0       0       0       0       0
0       0       0       0       0       0       0       0       0       0
0       0       0       0       0       0       0       0       0       0
0       0       0       0       0       0       0       0       0       0
0       0       0       0       0       0       0       0       0       0
0       0       0       0       0       0       0       0       0       0
0       0       0       0       0       0       0       0       0
time
1
C:\jdk1.6\bin>rrrr_
```

CLUSTER INDEX:



```
silu
simran
sinu
sigt
sigti
srin
subu
subu
sunil
surya
syio
tata
tata
tini
tinu
unni
unni
veni
wilington
wini
yeni
time
0
C:\jdk1.6\bin>_
```

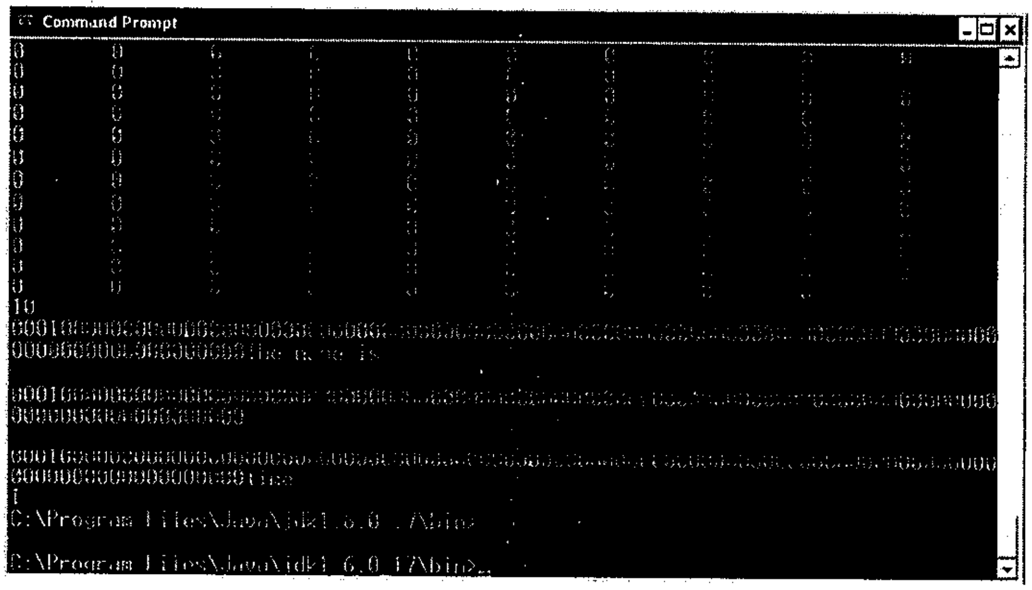
HASHBASED INDEX:



```
The values in count[[illegible]] is[illegible]
The value in count[5] is[illegible]
The value in count[4] is25[illegible]
The value in count[4] is2344
The value in count[5] is2345
The value in count[5] is2345
The value in count[7] is2347
The value in count[7] is2347
The value in count[8] is2348
The value in count[8] is2349
The value in count[3] is[illegible]
The value in count[[illegible]] is[illegible]
The value in count[[illegible]] is3[illegible]
The value in count[8] is34
The value in count[7] is345
The value in count[5] is3458
The value in count[2] is3458
The value in count[8] is3488
The value in count[6] is3498
The value in count[8] is3448
The values in count[[illegible]] is3558
5

C:\jdk1.6\bin>_
```

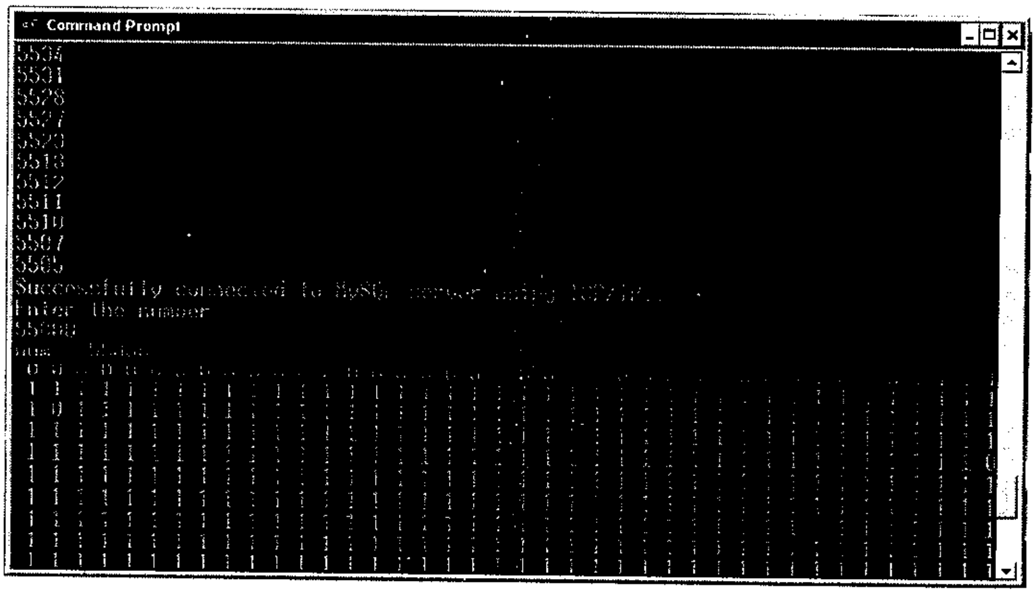
PROPOSED WORK:

SIMPLE BITMAP INDEX:



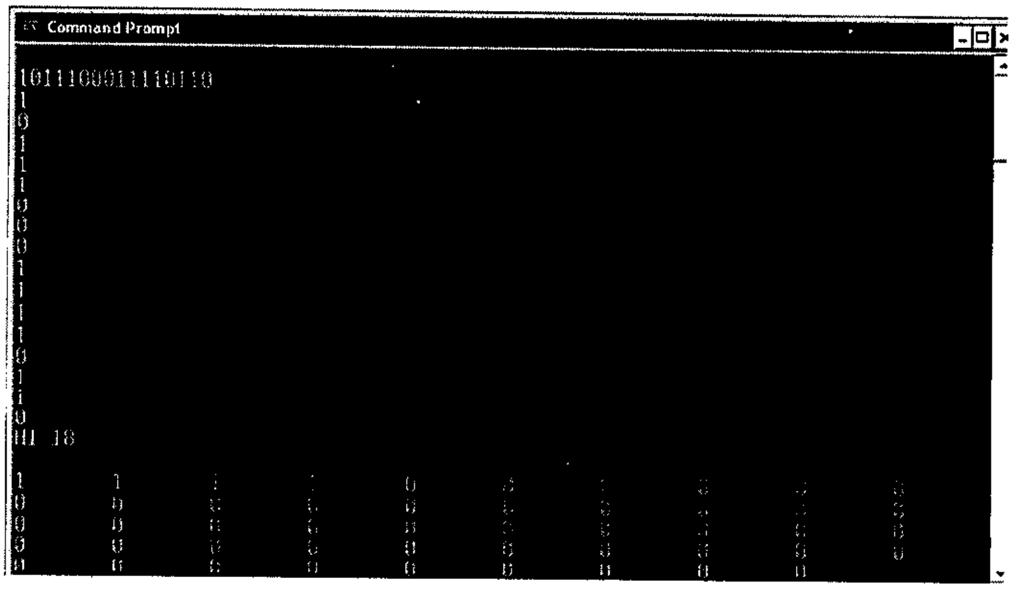
```
0         0         0         0         0         0         0         0         0         0
0         0         0         0         0         0         0         0         0         0
0         0         0         0         0         0         0         0         0         0
0         0         0         0         0         0         0         0         0         0
0         0         0         0         0         0         0         0         0         0
0         0         0         0         0         0         0         0         0         0
0         0         0         0         0         0         0         0         0         0
0         0         0         0         0         0         0         0         0         0
0         0         0         0         0         0         0         0         0         0
0         0         0         0         0         0         0         0         0         0
0         0         0         0         0         0         0         0         0         0
0         0         0         0         0         0         0         0         0         0
10
0001[illegible]
000[illegible]the [illegible] is

0001[illegible]
000[illegible]

0001[illegible]
000[illegible]time
1
C:\Program Files\Java\jdk1.6.0_17\bin>

C:\Program Files\Java\jdk1.6.0_17\bin>_
```

RANGEBASEDINDEX:



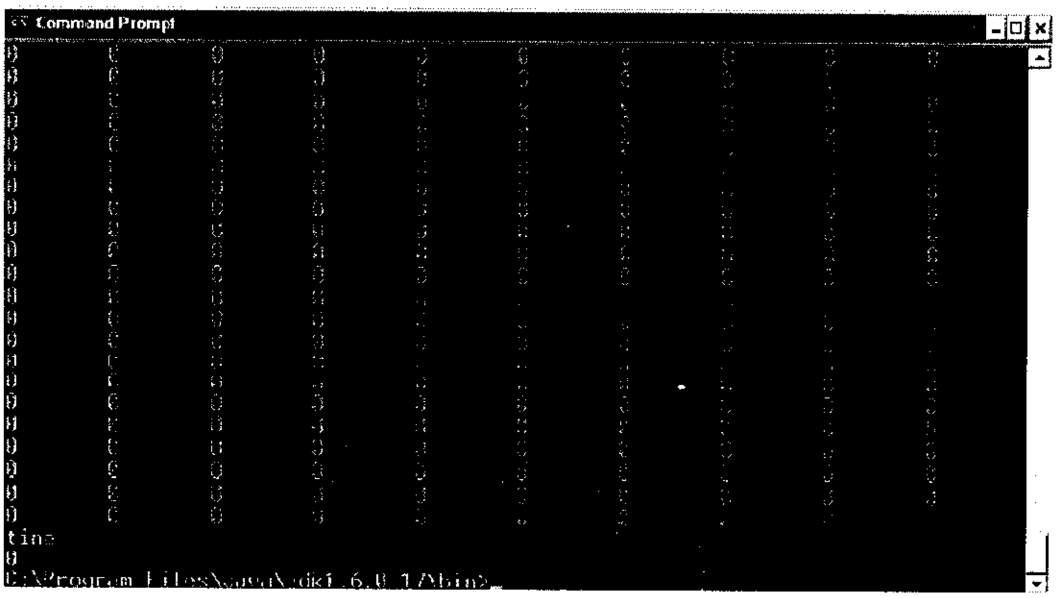
```
5534
5531
5528
5527
5520
5518
5512
5511
5510
5507
5505
Successfully connected to MySQL server using TCP/IP...
Enter the number
55000
[illegible]
 0 0 [illegible]
 1 1 [illegible]
 1 0 [illegible]
 1 1 [illegible]
 1 1 [illegible]
 1 1 [illegible]
 1 1 [illegible]
 1 1 [illegible]
 1 1 [illegible]
 1 1 [illegible]
```

BITSLICE INDEX:



```
1011100011110110
1
0
1
1
1
0
0
0
1
1
1
1
0
1
1
0
Hi 18

1        1        [illegible]  [illegible]  0        0        [illegible]  0        0        0
0        0        0        0        0        0        0        0        0        0
0        0        0        0        0        0        0        0        0        0
0        0        0        0        0        0        0        0        0        0
0        0        0        0        0        0        0        0        0
```

PROJECTION INDEX:



```
0          0          0          0          0          0          0          0          0          0
0          0          0          0          0          0          0          0          0          0
0          0          0          0          0          0          0          0          0          0
0          0          0          0          0          0          0          0          0          0
0          0          0          0          0          0          0          0          0          0
0          0          0          0          0          0          0          0          0          0
0          0          0          0          0          0          0          0          0          0
0          0          0          0          0          0          0          0          0          0
0          0          0          0          0          0          0          0          0          0
0          0          0          0          0          0          0          0          0          0
0          0          0          0          0          0          0          0          0          0
0          0          0          0          0          0          0          0          0          0
0          0          0          0          0          0          0          0          0          0
0          0          0          0          0          0          0          0          0          0
0          0          0          0          0          0          0          0          0          0
0          0          0          0          0          0          0          0          0          0
0          0          0          0          0          0          0          0          0          0
0          0          0          0          0          0          0          0          0          0
0          0          0          0          0          0          0          0          0          0
0          0          0          0          0          0          0          0          0          0
0          0          0          0          0          0          0          0          0          0
0          0          0          0          0          0          0          0          0          0
time
0
C:\Program Files\Java\jdk1.6.0_17\bin>
```

REFERENCES

[1] A.Abdullah, 'A brief introduction to data mining', August 2008.

[2] A. Aizawa, 'A method of cluster-based indexing of textual data', In Proceedings of the 19thinternational conference on Computational linguistics - Volume 1, International Conference On Computational Linguistics, Taipei, Taiwan, pp. 1 – 7, 2002.

[3 Chee-Yong Chan and Yannis E. loannidis. 'Bitmap Index Design and Evaluation'. University of Wisconsin-Madison.

[4] El Ghailani Maher and Dr. Hachim Haddouti,'Advanced database system and datawarehousing', Sql server magazine,October 2003.

[5] H. Gupta, V. Harinarayan, A. Rajaraman, and D. Ullman. 'Index Selection for OLAP', ICDE,1997.

[6] Herbert Schildt,'Java2:The complete reference', Tata McGraw-Hill publications,fifth edition.

[7] Matteo Golfarelli, Stefano Rizzi, and Ettore Saltarelli, 'Index selection for datawarehousing', In Proceedings of the 4th Intl. Workshop DMDW'2002 at Cease*02, Toronto, Canada, May 27, 2002. pp. 33-42.

[8] M. A. Pole , 'Indexing the data warehouse', SQL server magazine August 2008.

[9] Ming-Chuan Wu, Alejandro P. Buchmann. 'Encoded Bitmap Indexing for data warehouses'. DVS1, Computer Science Department, Technische Universitat Darmstadt, Germany.

[10] S. B. Davidson, 'Indexing, Sorting and Hashing', October 23, 2008, www.seas.upenn.edu/~cse330/docs/14-Indexing.ppt (visited date: may6, 2009).

[11] S. Sarawagi. (1997) 'Indexing OLAP data'. Data Eng.Bulletin 20 (1), pp. 36-43.

[12] S.Delmarco,'HashIndexes',January27,2006 http://www.fotia.co.uk/fotia/FA.03.Sql2KHashIndexes.01.aspx (visited date: may 6, 2009).

[13] V. Markl, R. Bayer, 'Processing Relational OLAP Queries with UBTrees and Multidimensional Hierarchical Clustering' In Proceedings of the International Workshop on Design and Management of Data Warehouses (DMDW'2000), Stockholm, Sweden, June 5-6, 2000