Spring 2012

An Algorithm for Data Reorganization in a Multi-dimensional Index

Urvashi Samaresh Nair
San Jose State University

An Algorithm for Data Reorganization in a Multi-dimensional Index

Writing Project

Presented to

The Faculty of Computer Science

San José State University

In Partial Fulfillment of the Requirement for the

Degree

Master of Science

By

Urvashi Samaresh Nair

May 2012

An Algorithm for Data Reorganization in a Multi-dimensional Index

By

Urvashi Samaresh Nair

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

_____

Dr. Chris Pollett

_____

Prof. Frank Butt

_____

Dr. Teng Moh

APPROVED FOR THE UNIVERSITY

ACKNOWLEDGEMENTS

I would like to thank my project advisor Frank Butt and my committee members Dr. Chris Pollett and Dr. Teng Moh for their guidance and direction on this project. I would also like to thank my husband Samaresh Nair and my family on helping and supporting during my project work.

ABSTRACT

AN ALGORITHM FOR DATAREORGANIZATION IN A

MULTI-DIMENSIONAL INDEX

by Urvashi Samaresh Nair

In spatial databases, data are associated with spatial coordinates and are retrieved based on spatial proximity. A spatial database uses spatial indexes to optimize spatial queries. An essential ingredient for efficient spatial query processing is spatial clustering of data and reorganization of spatial data. Traditional clustering algorithms and reorganization utilities lack in performance and execution. To solve this problem we have developed an algorithm to convert a two dimensional spatial index into a single dimensional value and then a reorganization is done on the spatial data. This report describes this algorithm as well as various experiments to validate its effectiveness.

Table of Contents

1. Introduction

1.1. Project Overview

The objective of this project is to develop an algorithm to speed up multidimensional spatial queries. We have focused on two dimensional indexes and developed a mapping from these two dimensional into single dimensional index. Our algorithm then does a reorganization of the data. In testing we have found our approach is faster than traditional multi dimensional indexing approaches. The main tools used in this project are DB2 and DB2 Control Center.

1.2. Report Overview

In this report, we describe the background work conducted before we came up with our reorganization algorithm. This background work formed the basis for the main project. At the start of the project we created the spatial indexes in the spatial database and experiments with index reorganization with the help of algorithms using DB2. Later on we explored and experimented with certain queries to justify the use of our algorithm and testing on its efficiency. This report ends with some ideas and innovation which could further develop the use of reorganization in spatial indexing.

2. Background Work and Theory

2.1. Spatial data and Spatial Grid Indexes

Spatial data is composed of sequences of coordinates each of which identify a location. For example, two-dimensional coordinates could specify an ***x*** and ***y*** or longitude and latitude values. We would notice that spatial data has a geographical location associated with it as it is in relation with points or data with more than one coordinates. There are distinct geometry types in spatial data such as point and polygon which we have used in our project [1].

Spatial data is used in research and everyday applications. The distances between two cities, postal codes of a city, number of students attending a particular school all have some element of spatial data. In the case of distance between two cities the data type for city location is represented in the form of point which has *x, y* coordinates. The data type polygon; a collection of points is used for postal codes of a city and number of students attending a particular school is referenced. Data types in spatial data are dependent on the data size and area coverage of the data. In general, spatial data is complex to represent and store in a database. When we have to query this spatial data we have to use spatial queries.

To have good query performance we need efficient index created on the column of the table which contains spatial data. A SQL query ran on a spatial table will have a significant performance improvement if an index is used. Spatial queries often use a so-called spatial grid index. Grid indexing is the technology used in IBM Spatial support for DB2 and it has been studied and further analyzed to understand the methodology to create the spatial grid index in our project [2].

Indexes improve query performance, especially when the queried tables contain many rows. If

appropriate indexes are created, and the query optimizer chooses to run using those indexes, we can greatly reduce the number of rows to process [3].

The following aspects of a grid index are helpful to understand: the generation of the index, use of spatial functions in a query, and how a query uses a spatial grid index.

A spatial grid index divides a region into logical square grids with a fixed size that you specify when you create the index. The spatial index is constructed on a spatial column in a spatial database table and it affects the processing and retrieval time taken to query the table.

2.2. Data Clustering

Clustering is the property wherein the data is classified into different groups depending on the similarity. It helps to retrieve the data which is organized with same or similar patterns with better and quick approach as same data clusters are located in same physical area. [4]

In single dimensional column of a database an index is a more efficient and faster approach to acquiring data from the table. Reorganization of data can take place with the help of an index. But in regards with two dimensional data column there is no DBMS infrastructure in place for having the data reorganized.

It is known that when spatial data is clustered per same or similar property, the data gets organized at same physical location in a table. [5] The prime reason for clustering data with similar characteristics is that the data can be accessed simultaneously. We create the spatial index on the column which has been created and generated by computing the value on spatial data element in the table. Data is reorganized based on the index which has been created on this new index.

We need to extend the table to contain the spatial data and have the clustering column added

which is generated by using a function which would compute the value for the element having the spatial data type [6]. In the case of DB2 spatial database there are built-in and defined functions which can be used to have a clustering column generated. The advantage of having the index created is we would be able to scan through the table in faster and reliable manner also we would need less number of I/O's per transaction or query. We observe that the index created on the table helps to limit the amount of data compared to a whole table scan.

2.3. Data Reorganization

When data becomes unorganized or fragmented, we perform data reorganization. We perform reorganization depending on the unused spaces, page gaps, I/O activity, and clustering present on the table. Data needs to be reorganized to have better performance but to have the reorganization tool or utility executed on huge data it would be time sensitive and a critical task.

We observe that there are common properties such as search and sorting used to retrieve the data from the database tables with help of an algorithm. For data that can be sorted in memory we can make use of Quick Sort algorithm and that which cannot be sorted in memory we use external sort-merge algorithm. In our project we observe that the spatial index created on the table sorts the data and spatial queries used help to retrieve the data or search the data in efficient manner.

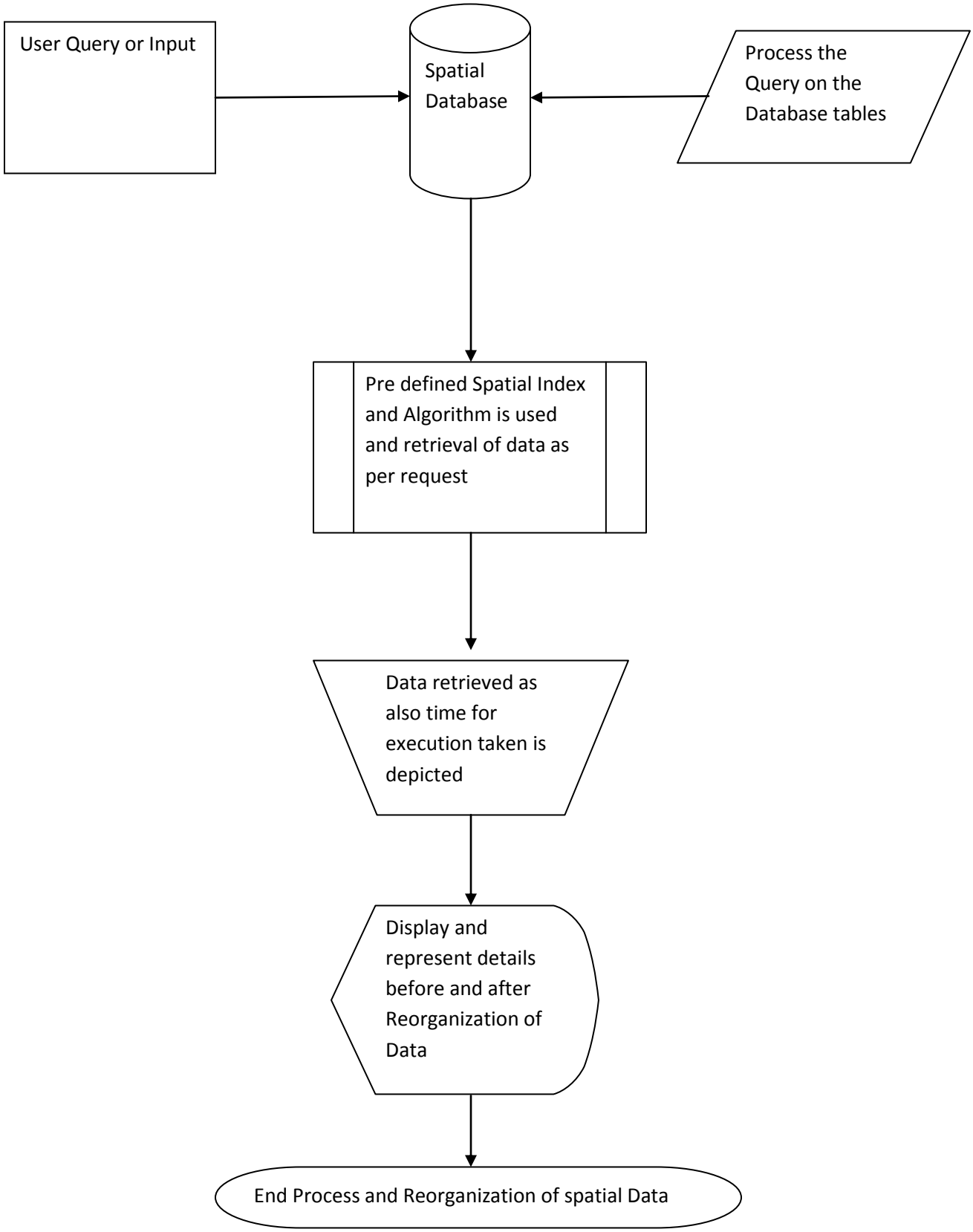
We use spatial data as the basis for our experiments and downloaded the data from census government website. Data types used are Point and Polygon. The technique for reorganization used is as follows:

- Calculate the centroids for both points and polygons.
- Consider a minimum bounding rectangle and calculate its centroid with help of function.

- Derive a new big integer value to store the centroid value.
- Store this big integer value into a new column of the table.
- Perform the reorganization of the table data based on the big integer value.
- Reorganization can now be done as the spatial data which had been two dimensional has been converted into a single dimensional value.

In the spatial database we create a table with this spatial data. The steps we perform are:

1. Create a table and import or insert data in the table.
2. Create a field which has spatial data present.
3. Alter or edit the table to include additional column or field.
4. Update the value of the field with help of user defined function.
5. Create an index on the column of the table.
6. Reorganize the table with help of index created.


User Query or Input
Spatial Database
Process the Query on the Database tables
Pre defined Spatial Index and Algorithm is used and retrieval of data as per request
Data retrieved as also time for execution taken is depicted
Display and represent details before and after Reorganization of Data
End Process and Reorganization of spatial Data

3. Implementation

We wanted to see the performance difference between two scenarios, namely, between our two dimensional spatial data clustering which is based on a non-spatial index, and a two dimensional spatial indexes. Details are mentioned on how our algorithm was deployed in on actual hardware and similarly for the two-dimensional spatial indexes.

3.1 Tools and Programming Languages

Our experiments were conducted using DB2 on the laptop with spatial data enabled and database created with spatial data.

3.2 Process Flow Diagram

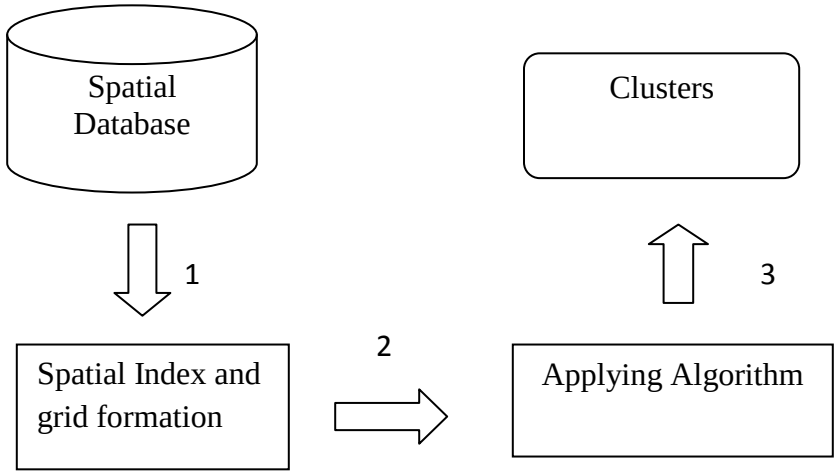


Figure 1: Process Flow Diagram

Step 1: Spatial Data is present in the spatial database. There are number of tables which have the spatial data.

Step 2: Spatial Index and grid formation is performed on the spatial data tables.

Step 3: Apply the algorithm and have the data classified into clusters.

- To perform two dimensional spatial data clustering based on a non two dimensional index which is non spatial (on a regular attribute of the table) we perform as follows:
    a. Reorganized the data on a regular attribute
    b. For timing did run the spatial query measuring the time taken.

  In our project we have three tables with different data sizes and have altered the table to include a column with non spatial data. Created an index on the column and reorganization of data is done with help of these indexes.

- To perform two dimensional spatial data clustering on a simulated two dimensional index (on a spatial attribute of the table) we perform as follows:
    a. Generated the big integer value using a user defined function that calculates the centroid value.
    b. Reorganized on the new big integer column.
    c. Run the same spatial query measuring the time taken (same as above for two dimensional spatial data clustering based on a non two dimensional index which is non spatial.)

After the reorganization was completed, we analyzed performance data with different data points depending on the following factors:

- Size of the table
- Number of rows returned
- Data type as Point or Polygon

3.3. Experiments to analyze the algorithm

Details on the test cases and data points on which experiments were performed are as follows: We created tables with the spatial data type of point. The data sizes of the test tables were of size ten thousand, hundred thousand and ten million data points. There were fields of data type integer and big integer added to the table. The update and the reorganization of the data had been achieved with index created on these two fields. The spatial query used was given different ranges to retrieve records.

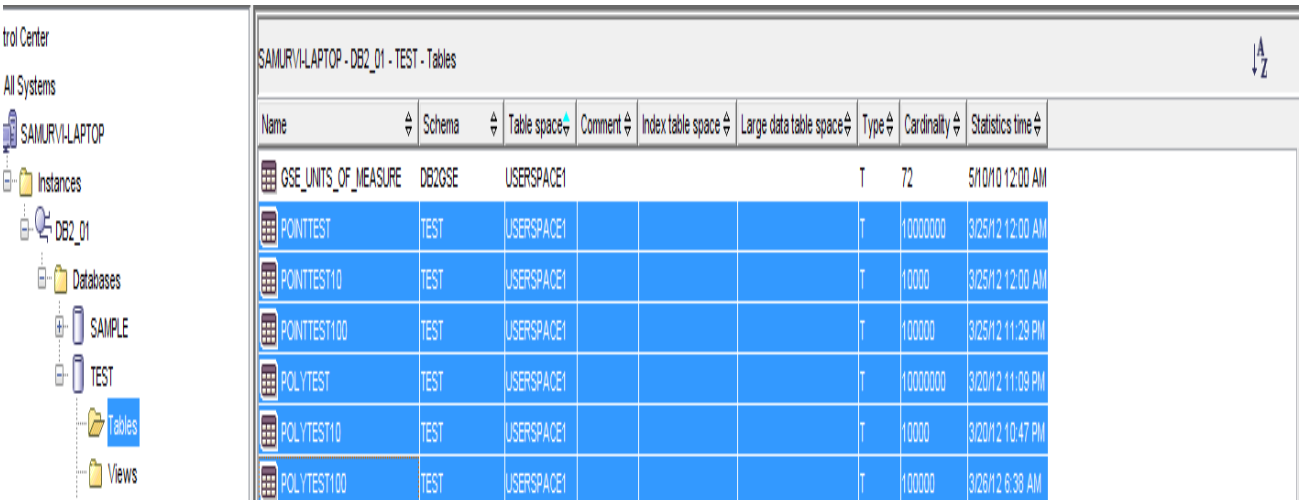


Figure 2: Tables created in Test Database

The access path in DB2 is determined during the bind process. It provides information on the indexes, its efficiency, and the shortest path that would be chosen by DB2 optimizer to retrieve or access the data from the database.

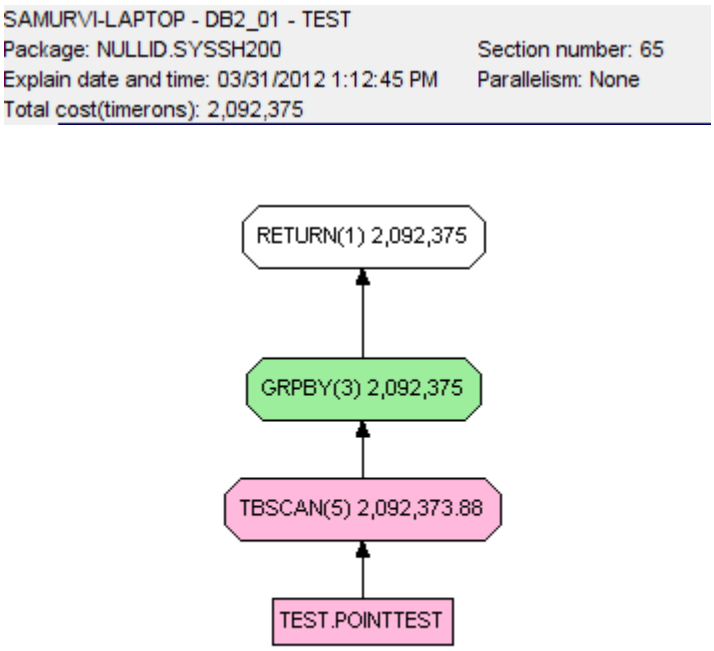


Figure 3: Access Path details for Point Test table

The approach taken to have the test cases and test results performed are as follows:

- Select the count (*) and time for envelopes window: We ran a query to find the time taken to retrieve the amount of records for different ranges or envelope sizes.

- Select the ST_TEXT of the polygon and also the time with the envelopes intersect and different ranges (windows): The query selected consisted of the ST_TEXT as the data in the tables is spatial data and we could have it run on different query ranges.

- 3 different window sizes and the count of records have been selected for the different tables, so that we can note that the amount of time and retrieval of data is different and not the same.

- Create index on non spatial data field and then reorganize the table on this index. Rerun the queries and note the time.

- Create index on spatial data field and then reorganize the table on this index. Rerun the queries and note the time.

The queries used were follows:

SELECT CURRENT_TIMESTAMP **AS** BEFORE_TIME **FROM** SYSIBM.SYSDUMMY1;

SELECT COUNT (*) **FROM** TEST."POINTTEST" AS C

WHERE DB2GSE.ENVELOPESINTERSECT (C.POINT,-0.032288913,-0.068835052, 0.065195868, 0.12184641, 0) =1;

SELECT CURRENT_TIMESTAMP **AS** AFTER_TIME **FROM** SYSIBM.SYSDUMMY1;

3.4. Test Result

ST_POINT

Spatial Table with point data type has a spatial index and normal index created. We would run a spatial query to obtain the time taken after reorganize on normal attribute data element and on spatial data attribute.

Table 1: Details on Time Taken and Count of records for Point data type table

Count of Records	Time taken after reorganize on Normal data attribute	Time taken after reorganize on Spatial data attribute	Total Number of Records in Table	Spatial Data Type
190	0.249	0.254	10000	Point
1650	1.7	0.402	100,000	Point
2010	2.02.581	1.35.885	10,000,000	Point

ST_POLYGON

Spatial Table with polygon data type has a spatial index and normal index created. We ran a spatial query to obtain the time taken after reorganize on normal attribute data element and on spatial data attribute.

Table 2: Details on Time Taken and Count of records for Polygon data type table

Count of Records	Time taken after reorganize on Normal data attribute	Time taken after reorganize on Spatial data attribute	Total Number of Records in Table	Spatial Data Type
280	0.224	0.214	10,000	Polygon
2460	10.954	1.785	100,000	Polygon
4100	1.13.913	1.16007	10,000,000	Polygon

4. Comparison with existing algorithms

Two common spatial indexing algorithms are K-Means and DBSCAN. We observe that an algorithm can be classified depending on the data which can be partitioned , hierarchical, and density-based. All the algorithms try to solve the clustering problem; Special structures are used to store the indexes such as Grid file, R-tree. [7]

Grid file: A two dimensional array is called **grid array** and one dimensional arrays are **linear scales**. [8] Spatial Grid Index uses the minimum bounding rectangle of the geometry. In general there are three grid levels or spatial index levels defined. They provide a significant decrease in processing time for multiple key queries but then they impose a space overhead concern. It is also difficult to select on partitioning ranges which are uniform. If there are frequent insertions or changes, then reorganize is expensive.

5. Computing Efficiency

We have created tables each for point data type and polygon data type. The time taken to run the same spatial queries with different data sizes is observed.

The following tables show the efficiency before and after usage of algorithm:

Table 3: Details on Time Taken and Count of records for Point data type table (with normal attribute)

Time Taken	Data points	Organized on Normal Attribute
0.249	190	YES
1.7	1650	YES
2.02.581	2010	YES

Table 4: Details on Time Taken and Count of records for Point data type table (with spatial attribute)

Time Taken	Data Points	Organized with spatial attribute
0.254	190	YES
0.402	1650	YES
1.35.885	2010	YES

Table 5: Details on Time Taken and Count of records for Polygon data type table (with normal attribute)

Time Taken	Data Points	Organized on Normal Attribute
0.224	280	YES
10.954	2460	YES
1.13.913	4100	YES

Table 6: Details on Time Taken and Count of records for Polygon data type table (with spatial attribute)

Time Taken	Data Points	Organized with spatial attribute
0.214	280	YES
1.785	2460	YES
1.16007	4100	YES

6. Conclusion

We have developed an algorithm to improve the efficiency of spatial indexes. We believe that our work can be extended to reorganizations depending on more than one field. The advantage of this algorithm is that as we have created a field in the spatial database which stores data in spatial form and an index on this field helps in managing the overhead cost and faster retrieval.

Clustering of the data in spatial database has made use of both non-spatial and spatial clustering indexes. [9] As the number of rows retrieved increases the difference in time taken to retrieve it is visible and significant. We can have more than two dimensional data reorganized once we know the formula to have the data reorganized. There would be no reason for us to not have the scope expanded.

We explored on the two dimensional index and closely studied the spatial indexes. The focus has been on algorithm for generating reorganization of two dimensional indexes so that the data is in clusters of relevant information. We were successful in having the two dimensional index translated or incorporated as single dimensional and organized depending on it.

References:

[1] IBM Infocenter DB2LUW , URL:

http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp?topic=/com.ibm.db2.udb.admin.doc/doc/c0005407.htm

[2] IBM DB2 V9 , URL:

http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp?topic=/com.ibm.db2.udb.admin.doc/doc/t0009675.htm

[3] IBM DB2 EPublications, URL : http://publib.boulder.ibm.com/epubs/pdf/dsnspk14.pdf

[4] Haralambos Marmanis; Dmitry Babenko (May, 2009) . Algorithms of the Intelligent Web by Manning Publications Co, 121-157.

[5] Garcia-Molina, Ullman, Widom Database Systems (June 2011) . The Complete Book, 2nd Ed. by Prentice Hall

[6] Thomas Kyte (July 2010) . Expert Oracle Database Architecture: Oracle Database 9i, 10g, and 11g Programming Techniques and Solutions, Second Edition by Apress, 425-493.

[7] Susan Lawson; Daniel Luksetich (Oct 2007) . DB2 9 for z/OS: Database Administration Certification Study Guide by MC Press

[8] Lilian Hobbs; Paul Tsien . Oracle Database 10g Release 2 Online Data Reorganization & Redifinition An Oracle White Paper , May 2005 URL:

http://www.oracle.com/technetwork/database/features/availability/ha-10gr2-online-reorg-twp-131644.pdf

[9] Knut Stolze- IBM Developer Works URL:

http://www.ibm.com/developerworks/data/library/techarticle/dm-0510stolze/