*International Journal of*
*Emerging Research in Management &Technology*
*ISSN: 2278-9359 (Volume-4, Issue-12)*

**Research Article** | **December 2015**

# Implementation of Hilbert Tree Algorithm for Applying Indexing Techniques in Berkeley DB

**Badal K. Kothari***
Research Scholar, Mewar University
Gangagrar, Chittorgarh, Rajasthan, India

**Dr. Ashok R. Patel**
Department of Computer Science,
Hem. North Gujarat Univeristy, Patan, Gujarat, India

***Abstract—***

***The growing interest within the field of data warehousing is turning into more and more crucial for decision makers to tack quicker and effective decision. On-line decision needs terribly short response time. There are several algorithms are used for various indexing techniques are on the market for bring this goal. In Multi-Dimensional databases the moist Indexing techniques that have attracted attention are Cube Indexing and Bitmap Indexing. In this research paper, our primarily target a way to improve existing cube indexing and storage engine which will allow us to enhance query resolution and optimization for obtaining faster response. For that we will attempt to implement the Hilbert R-Tree algorithm rules with the Integration of Berkeley DB and Existing Sidera Server Cube indexing.***

***Keywords— Data Warehouse, OLAP, Muti-Dimensional OLAP, Cube Indexing, Hilbert R-Tree, Berkeley R-Tree, Sidera R-Tree.***

## I. INTRODUCTION

Most Important problem in the field of DSS (Decision Support System) is the querying and accurate and efficient accessing of data in Multi-Dimensional Data stored in the Data Warehouse. An Online Analytical Processing (OLAP) DBMS sometimes known as a OLAP Storage Engine is responsible for how data of the data warehouse is stored effectively on disk and access quickly. The Data Cube is multidimensional data model that supports OLAP processing that allows viewing aggregated data from a number of perspectives. A cube consists of dimensions and measures. For a D-Dimensional space, {A1,A2, ---, Ad} we have O(2d) attribute combinations known as views or cuboids or group-bys.

The existing OLAP Storage Engine like Sidera Server employs several components is used to answer Multi-Dimensional OLAP Queries in a very efficient and effective way. For a D-dimensional fact table is associated with d dimensions, the current Sidera storage engine creates the R-tree indexed data cubes as a 2*2d separate standard disk files. Two separate files represent the R-tree indexing for cuboids in a D-dimensional cube. These simple files are not databases in any sense and cannot efficiently support DBMS / OLAP queries.

Berkeley DB provides a high-performance embedded database for Key data. Berkeley DB (BDB) stores arbitrary key pairs as byte array and supports multiple data items for a single key. BDB combines the power of a relational storage engine with simplicity of file system based storage.

In this paper, we will focus on the new approach that integrates the functionality Hilbert R-Tree for implement the Cube Indexing and access component of Berkeley DB into existing Sidera Server.

## II. HILBERT R-TREE

The performance of the R-trees depends on however sensible is that the algorithmic rule that clusters the data rectangles to a node. Hilbert R-tree curves visits every node during a grid just the once in non cycle form. For static and dynamic databases there square measure 2 differing kind of Hilbert R-tree is offered. For impose linear ordering on the Data rectangles; Hilbert curves used by the Hilbert R-Tree. Hilbert Curve are used to bring the bacon higher ordering within the sense of grouping similar data rectangles along, to attenuate the realm of the ensuring Minimum Bounding Rectangles of multi-dimensional objects within the node.

Below Figure-1 shows the Hilbert curve on a 2x2 grid, denoted by H1. To derive a curve of order i, every vertex of the fundamental curve is replaced by the curve of order two and three. Once the order of the curve tends to time, the ensuring curve may be a form of fractal, with a form dimension of two. The Hilbert curve are often generalized for higher dimensionalities.

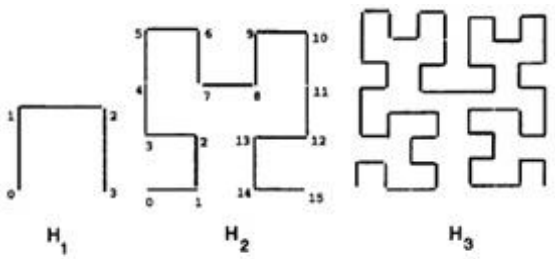



Figure-1: Hilbert Curves of order 1, 2 and 3

Figure-1 shows one such ordering for a 4x4 grid. For instance, the point (0,0) on the H2 curve incorporates a Hilbert value of 0, where as the point (1,1) incorporates a Hilbert value of 2. The Hilbert value of a rectangles must be outlined.

Hilbert R-tree structures shows that form (R,OBJ_ID) contains C1 as a leaf node, OBJ_ID as a pointer and R as a Real objects MBR of the object description records while form (R,ptr,LHV) contains Cn as non leaf node, R as a MBR which contains all child of Cn nodes,ptr as a child node pointer and LHV as a maximum Hilbert value from all the nodes enclosed by the R. Figure-2 shows rectangles represented in the Hilbert Tree.

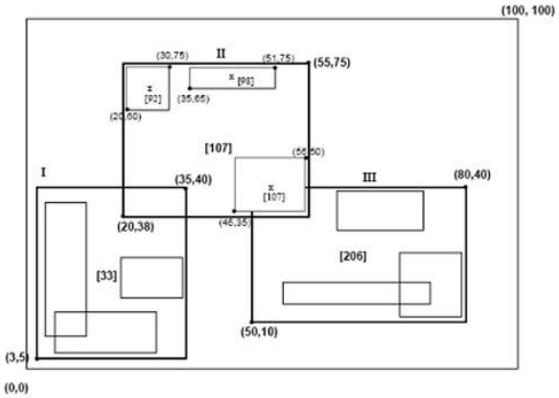



**Figure-2: Data rectangles organized in a Hilbert R-tree (Hilbert values and LHV's are in Brackets)**

## III. SIDERA SERVER: HILBERT R-TREE INDEXED CUBE

In Sidera explicit multi-dimensional indexing is provided by parallelized R-trees. The R-tree indexes are unit packed employing a Hilbert space filling curve. So, that arbitrary k-attribute varies queries more closely map to the physical ordering of records on disk. For every cuboids fragment on a node, the basic process is as follows:

Sort the data based upon the Hilbert sort order. Associate each of the n points records with m pages of size n/M. Write the base level to disk as cuboids name.hil.

↓

Associate each of the m leaf node pages with an ID that will be used as a file offset by parent bounding boxes.

↓

Construct the remainder of the R-tree index by recursively packing the bounding boxes of lower levels until the root node is reached.

The end result is a Hilbert-packed R-tree for each cuboids fragment in the system. The Hilbert packed R-tree is stored on disk as two physical files: a .hil file that houses the data in Hilbert sort order, and a .ind file that houses the R-tree metadata and the bounding boxes that represent the index tree.

### *A. Cube Indexing Integration.*

Sidera server uses the encoded fact table to generate the full cube as 2d views in a D-dimensional space. The Sidera indexing component described how the full cube can be materialized for a D-dimensional space. In other words, the indexed R-tree[14] for each cuboids is stored on disk as two Physical files: .hil file that contains the data in Hillbert sort order [17], and .ind files that contain the R-tree index metadata and the bounding boxes that represent the index tree. These files (.hil and .ind) are not databases and are not particularly efficient for OLAP quearies.

Berkeley provides four access methods B-Tree, Hash, Recno and Queue that perform very well in the context of the primary index, it is not sufficient in its current form to efficiently support the multi-dimensional queries that are executed by the Sidera server.

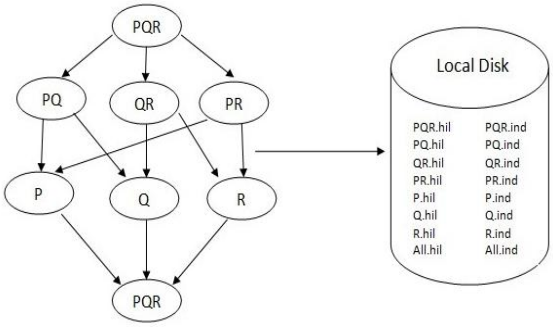



**Figure-3 : 3-dimenstional cube (PQR)**

Berkeley understands the notion of index / data combinations, it has no mechanism to directly support multi-dimensional R-trees.

The integration process (Berkeley DB and Sidera server) consists of combining the source code for the cube indexing with the code of Berkeley DB. After this integration, Berkeley DB can be used to create a Berkeley DB database with the Hilbert R-tree access method.

## IV. BERKELEY R-TREE MODEL

Berkeley supports the storage of many databases i.e. Berkeley Database Objects in one physical file. This physical file contains one master database supported by the B-Tree access method that has references to all the databases that are stored in the same file. Keys in this primary B-Tree are the database names that are stored in the physical file and the data of the primary B-Tree consists of the meta-data page number for each database name.

After the integration of the Berkeley and Sidera, Instead of 2*2d physical files, it will build the indexed cube as O(2d). Berkeley database objects for a D-dimensional fact table, one Berkeley database with the R-tree access method for each materialized cuboids.

Berkeley Db physical file contains a master database that has references to Hilbert R-tree indexed cuboids stored as Berkeley database in same file.

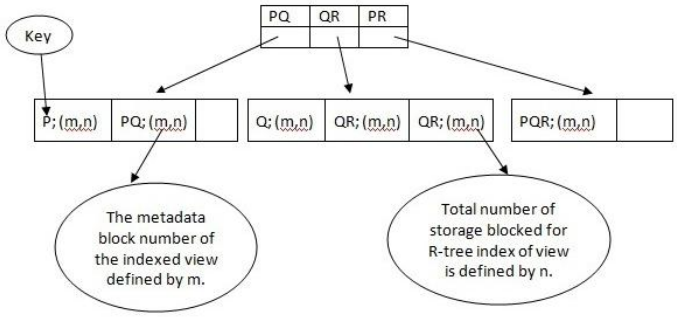



**Figure-4: Berkeley B-tree index that has references to all indexed cuboids stored in one Physical file.**

Keys in the master database are the indexed cuboids name, and the data contains two values: the indexed cuboids page meta-data number and the size in terms of number of blocks required by this indexed cuboids.

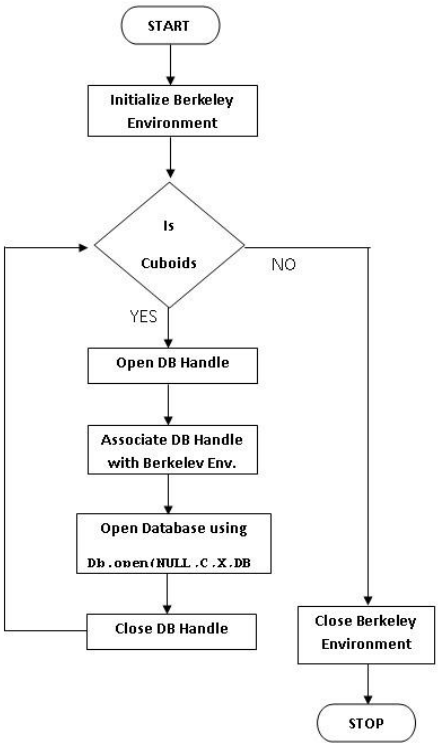
### *B. Berkeley Database: Hilbert R-Tree Indexed Cube*

Below algorithm represents, how the Hilbert R-tree indexed cube is stored as Berkeley database in one physical Berkeley database file.

***Input:*** *A set S of cuboids/cube name called C.*

***Output:*** *Hilbert R-tree indexed cuboids stored as Berkeley database objects in one physical file.*

***Algorithm:***



At beginning, create and open Berkeley database environment handle that encapsulates one or more Berkeley database objects. One for each indexed cuboids in the cube. Then, for each cuboids X in the cube, create Hilbert R-tree indexed view as a Berkeley DB database with the database type DB-RTREE in the open method. Berkeley DB open method with database type DB RTREE and a view X means that if the Berkeley database (X) does not exist then, first create it and store in it the Hilbert R-tree index corresponding to cuboids X. In the open method, the name of the physical file that will be used to back one or more Berkeley DB R-tree database will be the name of the cube.

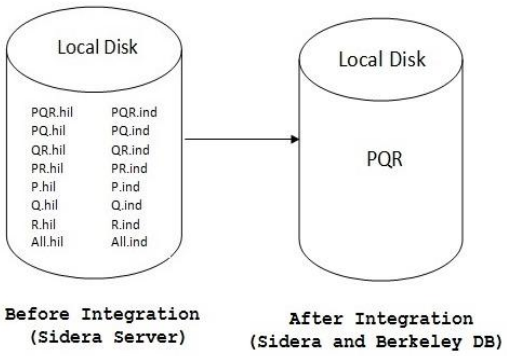



**Figure-5: Hilbert R-tree indexing of the 3-dimensional cube (PQR) before and after integration of Berkeley DB and Sidera Server.**

When we open a Berkeley DB database handle with a database type equivalent to DB-RTEE, a Hilbert R-tree indexed cuboids is created and stored in one Berkeley DB physical file that has the same name as the cube.

As noted, the indexed cube in Berkeley is represented only in one single physical file; however, it is represented in 2*the number of view in the current Sidera server. It will reduce the index cube construction time. The primary reason for this reduction is the need for only one physical file to store the index cube in the Berkeley supported Sidera. Storing the indexed cube as 2*2d physical files produces a significant amount of disk thrashing.

## V. CONCLUSIONS

In this paper, we have described the implementation of Hilbert Algorithm for applying Indexing Techniques in integration of Berkeley DB components with the Sidera server. This integration significantly enhances the existing Sidera storage engine. Specifically, Sidera now stores the Hilbert R-tree index in one Berkeley DB physical file. We have conceptually test the integration of the Sidera with Berkeley DB in terms of the index cube construction and query resolution. We have also compared the index construction for the cube in a single backend node before and after the integration of the codes in Berkeley DB.

Conceptually, our results support the integration approaches that have taken. Berkeley supported Sidera server is better than the old Sidera server and it significantly boost run-time query performance.

## REFERENCES

[1] Wikipedia, the free encyclopedia. https://en.wikipedia.org
[2] Badal K Kothari, Dr. Ashok R Patel, Cube Indexing Implementation Using Integration of Sidera and Berkeley DB, International Journal of Computer Engineering and Applications (IJCEA),Volume VII, Issue III, Part I, September (2014).
[3] Microsoft analysis services. http://www.microsoft.com/sqlserver/2008/en/us/ analysisservices.aspx.
[4] Oracle essbase. http://www.oracle.com/us/solutions/ent-performancebi/ Business – intelligence/ essbase/index.html
[5] Xml for analysis specification v1.1, (2002). http://www.xmla.org/index.htm
[6] Table. Query Optimization and Execution for MOLAP, pages 34-4478-87 (2011).
[7] Cwm, common warehouse metamodel,(2003). http://www.cwmforum.org
[8] Jsr-69 javatm olap interface (jolap), jsr-69(jolap) expert group, http://jcp.org aboutJavcommunityprocess/first/jsr069/index.html.
[9] E.Zimanyi E. Malinowski. Hierarchies in a conceptual model: From conceptual modeling to logical representation. Data & KNowledge Engineering, (2005).
[10] Todd Eavis, George Dimitrov, Ivan Dimitrov, David Cueva, Alex Lopez, and Ahmad Taleb. Sidera: A cluster-based server for online analytical processing. International Conference on Grid computing, high-performAnce, and Distributed Applications [GADA], (2007).
[11] Arnaud Giacometti, Dominique Laurent, Patrick Marcel, and Hassina Mouloudi. A new way of optimizing olap queries. BDA, pages 513–534, (2004).
[12] Ralph Kimball and J. Caserta. The Data Warehouse ETL Toolkit. John Wiley and Sons, (2004).
[13] Ralph Kimball and Margy Ross. The Data Warehouse Toolkit. John Wiley and Sons, (2002).