INTERNATIONAL JOURNAL OF RESEARCH IN COMPUTER APPLICATIONS AND ROBOTICS
ISSN 2320-7345

IMPACT OF DATA STORAGE TRENDS ON QUERY EXECUTION

Dr. Vandana Bhagat[1], Mr. Arindam Chatterjee[2]

[1] *Assistant professor, AIMS Institutes of Higher education, Peenya, Bangalore, vb11.vandana8@gmail.com*
[2] *mail4arindam7@gmail.com*

Abstract: - Data has become a vital part for all types of industries. Data can be made intelligent to take valuable decisions. This approach has raised a need for big amount of data to cover all the views of organization. When data is in large amount, managing it is not a easy task. It requires lot many things to concentrate. The major points to consider are data storage file format, data storage strategies and size of data to be accessed. This paper has discussed about all such concepts and inventions for efficient data storage. The query performance also depends on data retrieval approach. Several techniques are available to retrieve the data in efficient way. It also depends on the way data is stored, different types of indexes, size of data and number of seeks needed to fetch the required data. This paper also includes the basic data storage structure with its challenges and their possible solutions. In this paper several solutions and options are provided to help end user to decide the data storage strategy for efficient query execution.

Keywords*:* data organization, query processing, physical and logical data storage, data structures, data file formation

1. Introduction

The need of smart data storage technique is increasing day by day, because the value of data has increased and it is going on increasing. As the data has become very important to make the smart decision, all the organizations have started to maintain large amount of data to cover all the views of organization. Also the data gets stored to fulfil the requirements of different types of users in the organization. Inefficient data storage practices can raise the problem of unreliable results and ultimately unproductive decision which may be hazardous to the organization. There is a need to find optimal ways to store, analyse and remonetise the data over the time. Data storage strategies provides an industry insider's insight on how to properly scope, plan, evaluate, and implement storage techniques to maximize performance, capacity, reliability, and power savings. This paper mainly concentrates on the efficient data storage strategies and its effect on query processing. It covers multiple data storage and retrieval techniques for faster result generation. It is believed that, data storage strategies are responsible for query processing speed. [15]

2. Data Storage

Computer stores data in the form of electromagnetic or optical form, which can be accessible for computer processor. The methods to store the data vary greatly, but the basic concept is always same. Information is kept in such a way that it can be accessed again later. The social networks, internet of things, scientific experiments and commercial services play a significant role in generating a vast amount of data. [22]
Data storage technology basically manages I/O operations in a computer system. The storage is classified into two categories depending upon the usage and location of the data. Primary storage holds the data in randomly accessed memory and other built in devices such as processor's L1 cache. Secondary storage maintains the data

on hard disks, tapes and other devices which require I/O operations. Primary storage is much faster to access than secondary storage because of proximity of storage to the processor, while secondary storage may hold large amount of data compared to primary storage. [25]

2.1 Computer Memory Hierarchy

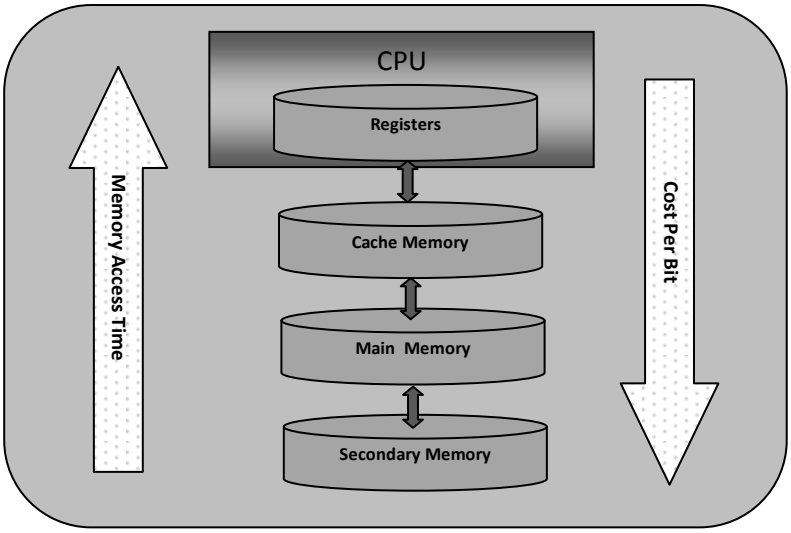


Figure 1: Computer Memory Hierarchy [21]

a. Registers: They hold the most frequently used data. They are very limited in number and are the fastest. They are often used by the CPU for performing arithmetic and logical operations, for temporary storage of data. [21]

b. Cache: They act as staging areas for a subset of the data and instructions. Storage is relatively slow than main memory. There are often two or more levels of cache as well. The cache at the topmost level after the registers is the primary cache. Others are secondary caches. L1 cache is accessed without any delay. L2 cache takes more clock cycles to access than L1 cache. L3 cache takes more clock cycles to access than L2 cache.[21]

c. Main Memory: It refers to the physical memory & it is one central storage unit in a computer system. It is relatively large & fast memory, used to store programs and data during the computer operations. [21]

d. Secondary Memory:
It often serves as staging areas for data, stored on the disks or tapes of other machines connected by networks.

2.2 Physical Storage Structures

In RDBMS logical data structure such as tables, indexes & views are independent of physical storage structure. Because of which physical data storage can be managed without affecting access to logical data structure. When user generate new database, multiple files gets generated physically like data files, temp files, control files, online redo log files and database. [20]

2.3 Logical Storage Structures

The logical units of database space allocation are data blocks, extents, segments, and tablespaces.[19] *Data blocks* are specific number of bytes of physical disk space. These are the smallest units of storage. *Extent* is a set of logically contiguous data blocks allocated for storing a specific type of information. *Segment* is a set of extents allocated for a specific database object or table. It is divided into different parts like data segment or index segment. Each segment belongs to one and only one *tablespace*. Therefore all extents for a segment are stored in the same tablespace.

Data storage effect on query processing

A programmer needs to understand the memory hierarchy, because it has a big impact on the performance. If the data required by the program is stored in a CPU register, then it can be accessed in zero cycles during the execution of the instruction. If stored in a cache, 1 to 30 cycles. If stored in main memory, 50 to 200 cycles. And if stored in disk tens of millions of cycles. This is obvious delay being introduced. The cost per bit is the least for the secondary memory, which is more at the primary memory level. This increases till the register level where it is the maximum. There is a trade-off between cost, access time, size and all this needs to be taken into account when designing the memory hierarchy for a computer system.
To manage big amount of data the traditional solution is to add more space or use external storage devices. This includes usage of mixed hardware, media, and protocols – flash and hard drives, direct-attached, SAN, NAS, cloud storage etc. But this solution further increases the problem of data access cost. Number of seeks to find required data may get increased. [5] The solution for this problem is to efficiently utilize available memory by optimized storage instead of adding new resources and managing them. Efficient storage solutions deliver overall improvements throughout the environment, which is rarely the case with expensive storage solutions.

3. Related Work

Lots of research work is happening on optimized storage as per the type of data and its requirement. In [18], Scholl et al. proposed a method for providing a logical relational view of data to the user while transparently storing a hierarchical clustering of related tuples as nested relations using a subset of the Non-First Normal Form (NFNF) model for query optimization. Their proposal achieves a result similar to column abstraction and super tuples.
In [4], Ailamaki et al. evaluate CPU and cache-related overheads of various data page layouts, including row- and column-oriented choices. The main focus of this research is a choice of PAX, which is row-column hybrid storage to achieve combined benefits of both the types.
Fractured mirrors [23] store two copies of relations, also referred to as hybrid data storage—one row-oriented and one column oriented—to provide better query performance than either storage choice can provide independently. The mirroring also provides protection against data loss in the event of disk failure.
The Bubba system [13] used a novel combination of inverted files and a “remainder” relation comprised of non-inverted attributes to store a relation. The inverted files are used as a data compression technique for attributes which contain redundant values. The inverted files are similar to a true column-oriented storage system, and capture the benefits of reducing disk I/O to improve sequential scan time
Keeping in mind the importance of these points the next section of the paper has discussed about the basics and evolution of it.

4. Basic Data structures

In the previous section we have observed that, how data storage strategy is affecting the result calculation, decision making and eventually business performance. Decades of research has been done to optimize data storage. Various advanced technologies are getting developed to face the challenges coming from big data generation and maintenance. All advanced data storage methods are basically developed on basic concepts of data storage technologies. This section contains all the basic methods used for optimized data storage and retrieval.
Data is stored in the memory in two basic ways. Relational data storage using tabular format and non relational data storage using document based storage. In any way it has to lead to efficient data retrieval. The basic method used to achieve it is indexing. It is proved by many scientists that sequential search takes long time to search the required data. Sequential search is efficient only in case of small sized database where it can work efficiently on unsorted data. Indexing can substantially increase the performance of search operation but the biggest constrain is, it requires the data in sorted order. Sorting such a large amount of data further increases overhead. This section explains about different methods of implementing efficient index on the relational database. Though each method becomes solution of one problem it further leads to another problem. We will discuss all such methods considering their pros and cons.

4.1 Binary search tree

Index is chiefly created using very traditional data structure i.e. Binary Search Tree. In such type of tree the data is placed in the sorted order where binary search can be applied. The main problem of indexing was to keep

the data in a sorted format. Binary search tree gives the solution to this problem. In this type of tree the data will be placed in the file in sorted format. It doesn't need to reprocess the data to keep it in sorted format.
Though binary search tree is good solution for maintaining sorted data, it faces other problem like increased in number of seeks. Searching, insertion, and deletion operations in a binary search tree is O(h), where h is the height of the tree. However, in the worst-case search, insertion, and deletion time is O(n), if the height of the tree is equal to n. Thus in some cases searching, insertion, and removal is no better than in a sequence. Thus, to prevent the worst case, we need to develop a rebalancing scheme to bind the height of the tree to log n [30]. The solution of this problem is achieved by AVL tree.

4.2 AVL Tree

AVL is the tree which is named after inventors Adelson-Velsky and Landis . AVL tree is a self-balancing Binary Search Tree (BST) where the difference between heights of left and right subtrees cannot be more than one for all nodes [32]. As large amount of data is getting generated every day, the solution which we are finding should consider the problems of big data. In such cases any index cannot be placed in primary memory because of its large size. Binary search tree has to invest more seek time to fetch every node in the path of data, because the data is placed in secondary memory. It is proved that, as the tree grows deeper and deeper number of seeks to access the data gets increases. In traditional binary search tree each node can hold only one record. In such cases for millions of records, the tree may grow for several levels which face the problem of increase in disk seek. [32]

4.3 Paged Binary Tree

AVL trees tackle the problem of keeping an index in sorted order cheaply. They do not address the problem regarding the fact that Binary Searching requires too many seeks. Paged Binary trees address this problem by locating multiple binary nodes on the same disk page. In a paged system, you do not incur the cost of a disk seek just to get a few bytes. Instead, once you have taken the time to seek to an area of the disk, you read in an entire page from the file. When searching a Binary Tree, the number of seeks necessary is log2(N+1). It is logk+1(N+1) in the paged version. [27]
There are multiple types of Paged binary tree like B Tree, B+ Tree and B* Tree. B-Tree is a self-balanced search tree with multiple keys in every node and more than two children for every node. It solves all the problems faced by AVL tree. Though B -Tree solves many challenges of increased seek operations, it further faces two problems:

1. Worst – Case Search Depth

B-tree has to follow some predefined rules like, each B-tree should be of order m, tree grows from bottom to top and minimum elements present in each node are m/2. [28] These rules can lead to generation of the B-tree whose maximum height has reached with all the nodes in the tree having minimum allowed number of descendents. In such situation the problem is, though the places for more elements are available in each node, it is not getting used which leads to increased height of the B-Tree. With increase in height, number of seeks gets increased which reduces the performance.
The solution for this is B* tree where the nodes are more saturated. In this tree the minimum numbers of elements are not half the order of three as in B-tree but higher fraction of maximum. The main aim of this type of tree is to achieve less height to get more efficient search.

2. Sequential file access

All types of trees face the problem of sequential data access. The recursive method has to be applied to achieve sequential file access. It increases the processing time. B+ tree is the best solution for sequential access problem. All the leaf nodes in B+ trees are interconnected in sequential order. Therefore user can directly search from one leaf node to another without recursion. B+-tree provides indexed sequential access. It can facilitate both types of access to the data as per the requirement of the user.

4.4 Hash

- Sequential searching can be done in o(n) access time. i.e. Number of seeks grow in proportion to the size of the file.
- B-Trees improves on this greatly, providing $O(\log_k N)$ access where k= measures of the leaf size (i.e. Number of records that can be stored in a key).
- Hashing can achieve O(1) access. (i.e. No matter how big a file grows , access to a record always takes same small no. of seeks.).

Hash function **h(K)** is like black box that transforms a key **k** into an address. The resulting address is used as the basis for storing and retrieving records. [35]

5. Impact of Basic data structure on data storage and retrieval

All the above types of data structures are nothing but different ways of index implementation. Each type has its own pros and cons. But the overall storage strategy makes big impact on data retrieval. The size and type of index plays vital role considering following points.

- The longer the index, the lesser number of values that can fit in a node, and hence the more the height of the B+tree.[1]
- The more the height of the tree, the more disk accesses are needed.[1]
- The more the disk accesses the less the performance.[1]

Therefore it is observed that the size of the index values have a direct impact on performance.

5.1 Data Organization & its types:

Data organization is a method of classifying and organizing data sets to make them more useful. Organizing, Re-ordering or analyzing the arrangement of data items in a physical record is part of data organization. Organizations produce the data in structured or unstructured format. Businesses adopt data organization strategies in order to make better use of the data assets that they have. [33] Storage efficiency is the process of storing and managing data while consuming the least space possible and ensuring optimal performance.
Innovators in storage efficiency are focused on delivering more capacity and performance with fewer disks, without sacrificing performance or data availability. Storage efficiency is about technologies such as thin provisioning, data deduplication, Snapshot copies, Flash, cloning and using technologies to mix in SATA to store not only secondary but also primary data. Storage efficiency is important because companies want to invest their budgets into applications, tools they don't have and ways to help the business grow, not disk. [16] [34]

5.2.1 Data deduplication

Deduplication technology shrinks the data footprint by removing redundant data at the file or sub-file level. It's most common use is with backups and archives, but it's increasingly gaining acceptance with primary storage.

5.2.2 Compression

Data deduplication eliminates redundant data. Compression reduces the size of every piece of data based on algorithms. Compression can be done standalone or in conjunction with data deduplication.

5.2.3 Snapshots

Snapshots are point-in-time copies of files, directories or data volumes that are especially helpful in the context of backups. Some systems save space by copying only the changes and using pointers to the original snapshot. [16]

5.2.4 Unified Storage

Unified storage refers to a single storage system that can support both file access and block access. That means the array can be utilized for file systems and storing file data. By consolidating SAN and NAS functions onto one storage array, these functions can share pooled resources, spread efficiencies across both and require less management.

5.2.5 Storage Virtualization

Storage virtualization works on the concept of thin provisioning. Thin provisioning is the act of using virtualization technology to give the appearance of more physical resource than is actually available.

5.2.6 Cloning

Cloning technologies provide files in seconds using advanced storage techniques instead of traditional copies, offering significant advantages. Clones are exceptionally fast. While traditional way of copying can be time consuming however even the largest volumes can be cloned in a matter of seconds. A clone uses a small amount of space for metadata, and then only consumes additional space as data is changed or added. This greatly aids operation efficiency. [31]

5.2.7 Use of Flash Storage

Flash storage is a type of storage which is designed to electronically secure data. Flash achieves a transformational shift in computing by eliminating seek time with rotational delay. The response orders of magnitude provided by Flash is faster than spinning disk. An *All-Flash Array* is data storage which encloses several Flash memory drives. All-Flash storage contains no moving parts, which means significantly less heat generated, less power utilized, and less maintenance. [17]

6.2.8 Tiering Storage

Aged data becomes less accessed which can be stored on high-performance drives in storage device. Tiering method of storage stores the data according to the relative demand of the data which eventually increases efficiency and lowers the cost. This method allows to store, [17]

- Low-priority data containing rarely accessed information on higher latency equipment that uses less energy.
- High-priority data which is immediate demand is stored on low latency storage equipment that consumes the more energy

The larger the capacity and the slower is the operating speed, generally more efficient its energy use.

6.2.9 Hybrid storage

Hybrid DBMS is invented to achieve combined advantages of multiple DBMS. The hybrid database is of different types depending upon the strategy used to make them hybrid and types of databases combined together. Available types of Hybrid databases and their characteristics are as follows,

- *HyPer* : In-Memory hybrid , mirror/replica hardware-supported virtual memory management [24]
- *Hyrise* : In-Memory DB-Systems [9]
- *Vertica's Flex Store*: fine-grained hybrid [12]
- *Alchemydatabase*: Hybrid RDBMS/NOSQL [3]
- *GreenPlum* : Open source (Map Reduce) Not Relational Database [14]
- *Ingres's Vectorwise* (Ingres' is named as Actian) : PAX type . [10] This was a open source db earlier but was commercialized from June 2010.
- *PostgreSQL* : fine-grained hybrid , object-relational database management system, Not Open Source [26]
- *Aster's nCluste* [7]
- *Firebird*: Open source SQL RDBMS that runs on Linux, Windows, and a variety of Unix. Hybrid OLTP and OLAP applications. [11]
- *Apache Cassandra:* column-group kind of NoSQL DBMS(Does not support ACID) [6]

6.2.10 Document Based storage

A document-oriented database is a designed for storing, retrieving, and managing document-oriented, or semi structured data. Document-oriented databases are one of the main categories of NoSQL databases. Some existing document based databases are MongoDB, Cassandra, CouchDB, Terrastore, RavenDB, OrientDB, ThruDB etc. [8] Application generated in object oriented programming language contains different types of objects often having many optional fields. Even though objects are of same class, look different. Document store allow different types of documents in single store, with optional fields. But this type of storage faces the problem of data redundancy or incorrect data insertion. This is because the schema is not fixed therefore constrains can be generated. I.e. in relational database constrain NOT NULL insist the end user to enter the data if it is mandatory but it can be checked in document-oriented database. [8]

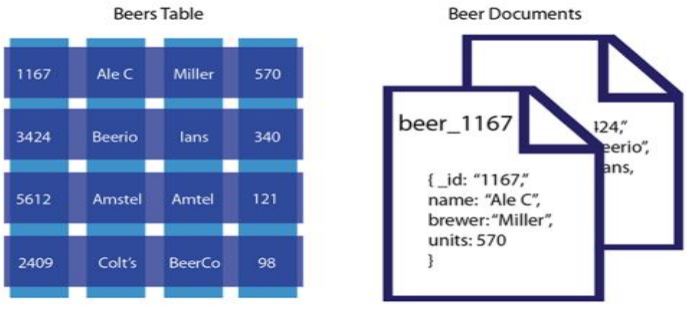


Figure 2: Relational database and Document-Based Database

7. Data Storage Strategies for selecting efficient data storage

Organization of data in the most efficient, scalable and organized fashion is important for valuable data access [29]. The data storage strategy may get changed as per the usage of data. Commonly 3 types of data storage formats are used by many systems like ASCII, Binary and XML. Each type has strengths and weaknesses in different area. This section has explained about these basic data storage strategies with their strengths and problems. It also explains how to decide the efficient data storage strategy for better performance. Table no 1 has listed diverse types of file formats and their characteristics.

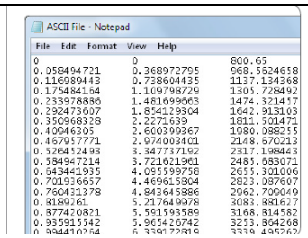
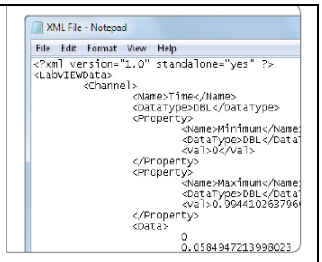
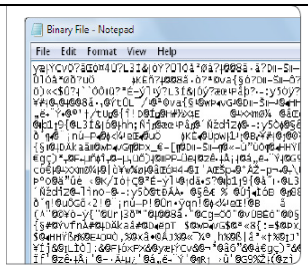
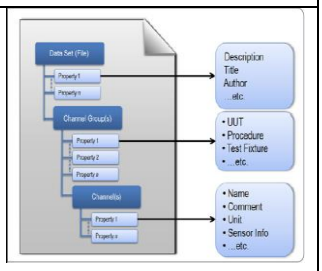
<table>
<tr>
<td>ASCII Files
Easy to exchange but can be too slow and large for many applications</td>
<td>  </td>
<td>XML Files
Helps to define complex structures but are significantly larger and slower than other formats</td>
<td>  </td>
</tr>
<tr>
<td>Binary Files
Beneficial in high-speed, limited-space applications but can cause exchangeability issues.</td>
<td>  </td>
<td>TDMS Files
TDMS is a binary-based file format, so it has a small disk footprint and can stream data to disk at high speeds.</td>
<td>  </td>
</tr>
<tr>
<td colspan="4">Database Files
Database files are composed of a series of tables, built using columns and rows, and information may or may not be linked between tables.</td>
</tr>
</table>

Table No. 1: Data Storage Format

From the following summary table no.2, it is proved that the TDMS file format combines the benefits of several data storage options in one file format. The comparison is done considering multiple features of data storage format like exchangeable, small disk footprint, searchable, inherent attributes, high-speed streaming and NI platform support.

	ASSII	Binary	XML	Database	TDMS
Exchangeable	✓		✓		✓
Small Disk Footprint		✓			✓
Searchable				✓	✓
Inherent Attributes			✓		✓
High Speed Streaming		✓			✓
NI Platform supported	✓	✓	✓	✓	✓

Table No. 2: Features of various Data Stooge Formats

Case study

It was observed by researchers [2] and program developers that data storage strategy shows big impact on performance of latest smart phones and tablets. The article published by Tom Simonite contains the summary of different findings of multiple researchers. According to this article the storage in phone has a bigger effect on apps in smart phones. The biggest bottleneck with smart phone apps specially Facebook and Google Maps which require big amount of data to be analyzed for decision making. [2] The testing was done using different memory cards in the mobile. Though it is believed that android apps are basically rely on internal memory, but for the experiment the researchers tweaked the operating system to rely on removable memory card.

It was observed that,
1. Gmail apps launched more than three times faster running on the best performing memory card than on worst.
2. The Twitter app launched more than twice as fast with faster storage.
3. The Android Web browser was tested by asking it to navigate to and load 50 top Web pages one after the other. It was around three times faster in the best case than in the worst.

These observations may show different results with different operating systems used in smart phone. This is because each operating system stores and accesses the data in different way.

8. Observation:

From the above study and various observations it can be analysed that, different data formats and data storage strategies shows remarkable impact on the query performance. From the comparative study of different data storage formats it was observed that TDMS file format gives combined benefits of several data storage options. Other than data storage file format, data storage strategies also affect query performance. This paper has shown different data storage strategies for improved query performance. Each storage strategy has its own strengths and weaknesses. For every weakness of one strategy, other strategy becomes the solution. Many inventions took place in data storage strategies and still more ideas are coming up. As per the user's requirement, the data storage strategy differs. It is observed that with a specified type of storage, data access method also gets changed.

9. Conclusion:

It can be concluded that to achieve the best performance from stored data, the organization of data has to be concentrated from basics. It includes how the data gets stored logically and physically in the computer system. The data should be stored in selected file format and storage strategy. The data access depends on the place, the type of data , the file format and subsequent size of the file. It also depends on data storage strategy and the amount of seek required to fetch the required data. There are chances of improved query performance if developer extensively studies the data storage and retrieval strategy.

10. REFERENCE AND CITATIONS

[1] Ovais.tariq, July 14, 2011, Understanding B+tree Indexes and how they Impact Performance.
[2] Tom Simonite, February 20, 2012, How Data Storage Cripples Mobile Apps
[3] Alchemy Database: A Hybrid RDBMS/NOSQL-Data store, [Online] , Available: https://code.google.com/archive/p/alchemydatabase/
[4] Anastassia Ailamaki et al. Data page layouts for relational databases on deep memory hierarchies. VLDB J., 11(3), 2002
[5] Andrew Flint, January 12, 2016, io fabric, Blog: Why Storage Efficiency Matters
[6] Apache Cassandra, (2016), [online], Available: https://en.wikipedia.org/wiki/Apache_Cassandra
[7] Aster Data Systems, (2016), [Online], Available: https://en.wikipedia.org/wiki/Aster_Data_Systems
[8] Comparing document-oriented and relational data, https://developer.couchbase.com/documentation/server/3.x/developer/dev-guide-3.0/compare-docs-vs-relational.html
[9] Dr. h.c. Hasso Plattner, (2016), "Enterprise Platform and Integration Concepts", [Online], Available: http://hpi.de/plattner/home.html
[10] Electronic Software Distribution,[Online], Available: http://esd.actian.com/product/Vectorwise/2.5.
[11] Firebird Features, [Online], Available: http://www.firebirdsql.org/en/features/
[12] FlexStore and the rest of Vertica 3.5, (2009),[Online] Available: http://www.dbms2.com/2009/08/04/flexstore-and-the-rest-of-vertica-35/
[13] George Copeland et al. Data placement in Bubba. In Proceedings of the ACM SIGMOD International Conference on Management of Data, Chicago, IL, June 1988. ACM Press
[14] Greenplum is going hybrid columnar as well , (2009),[Online], Available: http://www.dbms2.com/2009/10/14/greenplum-hybrid-columnar/
[15] Hubbert Smith, March 28, 2011 , Data Center Storage: Cost-Effective Strategies, Implementation, and Management, Auerbach Publications , Reference - 368 Pages - 63 B/W Illustrations ISBN 9781439834879 - CAT# K1157
[16] IP Pathways , May 2012, Storage Efficiency and The Rapidly Growing Data Center Footprint

[17] Kurt Marko, February 2012, Better Management of Data Storage
[18] Marc H. Scholl et al. supporting flat relations by a nested relational kernel. In Proceedings of 13th International Conference on Very Large Data Bases, September 1-4, 1987, Brighton, England
[19] Oracle help center, Logical Storage Structures https://docs.oracle.com/cloud/latest/db112/CNCPT/logical.htm#CNCPT004
[20] Oracle help center, Physical Storage Structures, https://docs.oracle.com/cloud/latest/db112/CNCPT/physical.htm#CNCPT003
[21] Ramnath, July 2016, Memory hierarchy with examples, ques10
[22] Randall Panduren, What is Data Storage? - Definition & Technologies
[23] Ravishankar Ramamurthy et al. A case for fractured mirrors. In Proceedings of the 28th Internation Conference on Very Large Data Bases, Hong Kong, China; August 20-23, 2002
[24] SAP HANA, (2010), [Online], Avaialble: https://en.wikipedia.org/wiki/SAP_HANA
[25] Searchstorage techtarget, December 2016, data storage
[26] Simon Riggs , (2010), "Hybrid Row/Column Data stores",[Online] , Available: http://database-explorer.blogspot.in/2010/01/hybrid-rowcolumn-datastores.html
[27] http://www.site.uottawa.ca/~nat/Courses/DFS-Course/DFS-Lecture-10/tsld008.htm
[28] http://www.site.uottawa.ca/~nat/Courses/DFS-Course/DFS-Lecture-10/tsld021.htm
[29] http://www.uio.no/studier/emner/matnat/fys/FYS3240/v12/undervisningsmateriale/daq/how_to_choose_data_storage_format.pdf
[30] https://www.cs.purdue.edu/homes/ayg/CS251/slides/chap7b.pdf
[31] https://www.dellemc.com/en-us/storage/discover-flash-storage/definitions.htm
[32] https://www.geeksforgeeks.org/avl-tree-set-1-insertion/
[33] https://www.techopedia.com/definition/30624/data-organization
[34] http://searchstorage.techtarget.com/feature/Data-reduction-techniques-for-better-storage-efficiency
[35] https://www.geeksforgeeks.org/hashing-set-1-introduction/

Author Biography

Dr. Vandana Bhagat – M.C.A., Ph.D. in sql query optimization and data storage techniques.

Arindam Chatterjee –M.Sc,PGDMRA