IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-7, Issue-4, April 2019 E-ISSN: 2347-2693

Exploring cloud based NOSQL Services

Mirza Zainab [1*], Savita Shiwani [2*]

[1,2] (Computer Sc. & Engg), Jaipur National University, Jaipur, India

[*]*Corresponding Author: mirza_zainab@yahoo.com*

DOI: https://doi.org/10.26438/ijcse/v7i4.686691 | Available online at: www.ijcseonline.org

Accepted: 16/Apr/2019, Published: 30/Apr/2019

***Abstract*—** Many times Relational Databases fails due to lack of support for handling the vast unstructured semantics of Big Data .Applications generating Big data are not efficient with relational databases in term of storage Space, database load time , query run time and application flexibility . When to use NOSQL data bases and which NOSQL database is good choice among the available NOSQL DB like Key-Value store, document store, column oriented DB, Graph DB or time Series DB needs to be evaluated carefully to get optimal performance in term of storage and query execution. This paper is about the Cloud based NOSQL –columnar oriented Database performance evaluation along with other NOSQL services like DynamoDB. Many prominent cloud providers like Google Cloud, AWS cloud and Microsoft azure Cloud they all provide NOSQL data bases as a service. These Databases are schema less, ACID free and flexible with full governance support for specific application need. This paper mainly focuses on cloud based column oriented data base performance with AWS Redshift.

***Keywords*—** NOSQL , AWS Redshift , AWS Athena , Columnar Databases, Key value DB

I. INTRODUCTION

Facebook, Yahoo, Google they all are relying on NOSQL databases for many services which is provided by them. Till 2005 - 2007, almost every internet application that hosted data creation, updations, deletion and display services i.e in all data management services harnessed their data handling strengths from SQL based RDBMS (Relational Database Management System). SQL is the standard language that is supported by such RDBMS for the basic CRUD (Create, Read, Update, and Delete) operations as well as the advance operations that consists of JOINS, TRIGGERS, Transactions, etc. Thus SQL technology was the most popular and widely used database technology at that time. But, since 2005 the size of data began to grow on an unexpected pace due to the advent of technology. Sources of data generation multiplied in numbers, playing the role of one of the major contributors to increasing size of data. This turning point in data technology, lead to the introduction of Big Data. The primary obstacle was that SQL was falling inefficient in terms of handling data that was increasing beyond its handling capacity. Although its capacity was stretched to occupy big data, but this put forth a lot of constraints on the actual functioning of SQL. One of the major issue was not only the size but also the widening of variety. Big Data constitutes the data that spreads in no one domain, but multiple domains. Data gradually began to lose structure and extensive applications endorsing image files, video files made it more difficult to store the actual files in directory file system and store their respective paths in the database. This proved cumbersome when retrieval operations came to light. As a counter attack to these issues, already introduced in 1998 NoSQL was reintroduced in early 2009 as a concept of database systems that are non - relational and did not expose the SQL. NoSQL functions at the cost of ACID transactions, but it provides extensible scalability, partitioning and distributed data storage.

This paper evaluates AWS cloud based services for data analytics and business intelligence i.e. AWS Redshift and AWs Athena.

II. RELATED WORK

Jagdev Bhogal and Imran Choksi in their IEEE publication “HANDLING BIG DATA using NoSQL”(published in 29th International Conference on Advanced Information Networking and Application Workshops, 2015) introduced primary and quite significant comparisons between the NoSQL and SQL systems in the big data context. NoSQL (abbreviation for not only SQL) is the emerging technology in data management domain and serves as a viable alternative to SQL based systems. The fact that is highlighted in this publication is that the major problem to solve is not; how NoSQL can replace SQL, but how NoSQL and SQL can be used in a complementary

fashion together. One needs to understand the requirements of data structure and accordingly decide to opt for whether SQL or NoSQL. The author demonstrates the significant trade-off between three major requirements of data management paradigm name Consistency, Availability, virtually infinite scale out. Although data stored in cloud based NoSQL systems is available anytime, anywhere and also they can be scaled on in infinite (preferably huge) scaling magnitude, but NoSQL systems trade-off with consistency and ACID transactions. Many times Big Data cannot be handled using relational databases i.e. SQL. The author demonstrated these peculiarities by implementing a small prototype application using both SQL and NoSQL. For SQL, Oracle APEX was considered and for NoSQL, the document-oriented MongoDB was selected. The author illustrated this difference by comparing two different visual schema definitions of the same application. One was the fixed schema to be implemented in SQL and another was the Document Schema to be implemented in MongoDB. When technically analyzed, the author attributed a fact why NoSQL provides better performance is that NoSQL schema can reduce the hassle of multiple joins and complex queries.
Normalizing the schema definition in SQL provides clean and segregated structure, but this also requires a number of joins and complex nested queries to retrieve even small amount of relationally linked data, on other hand NoSQL being of no fixed schema and flattened tables , can easily retrieve data with simple function calls (for eg: in MongoDB). The author concluded by suggesting applications of NoSQL in strategic business data and IoT based sensor networks that are nothing but the real-time data. Applications of referential integrity and which possess no concrete structure are more preferred to be handled by NoSQL systems. Dynamic data model applications are most suitable for NoSQL.

Yishan Li and Sathiamoorthy Manoharan in the IEEE publication “A performance comparison of SQL and NoSQL databases” make an effective comparison between the existing SQL technology for handling big data and the rapidly emerging NoSQL technology. The author puts forth the idea that not all NoSQL base database management techniques are as efficient as SQL but also attributed to the fact that SQL lacks some major features that NoSQL provides. In this publication, the author compares the key - value implementations of both NoSQL and SQL databases. The comparison is based on four major data management operations viz. Read, write, delete, retrieval. The author provides introductory information about the initial implementations of NoSQL technology namely Google’s Big Table and the Amazon’s DynamoDB. The author further provides information about the various existing implementations of NoSQL, among them are MongoDB, Hypertable, Apache CouchDB, Apache Cassandra, RavenDB and Couchbase. The author has compared above NoSQL implementations with the Microsoft’s SQL Server Express using an experimental setup to test the time taken by each of these implementations in performing operations like Instantiate, Read, Write, and Delete. Also, retrieval of the data is considered. For instantiate operation, the RavenDB NoSQL implementation ranks fastest while SQL Express ranks slowest. For reading operation, the Couchbase and MongoDB perform the best while RavenDB performs the worst. SQL Express ranks on the third positions in this comparison. For a write operation, Couchbase and MongoDB again bag the top 2 positions, while RavenDB and CouchDB finish at second to last and last respectively. SQL Express makes it to the fifth position on this run. For a delete operation, the results are quite similar to that of the results of the read operation. Couchbase, MongoDB and SQL Express prove themselves the front runners while RavenDB and CouchDB take the last positions. The maximum number of records that are read, written and deleted as well as fetched are 100000 records. In the final test i.e fetch all records operation, SQL Express ranked the fastest while other NoSQL implementations did well. CouchDB still was the last rank holder while Couchbase was excluded in this test as there are no APIs available for fetching all the keys. The author concludes by stating that NoSQL is peculiar because this technology is optimized for its key - value implementations. While SQL is not yet optimized. The author asserts that there are a wide variety of NoSQL implementations available and existing for applications. One can easily observe how different NoSQL implementations perform best for respective operations and worst for respective operations. The overall performance of MongoDB, a document-oriented implementation of NoSQL performed the best, if the inability of Couchbase to fetch all records is considered; otherwise Couchbase is the front rank holder. The author also argues that these database comparisons and standings may not hold true when these are tested for more complex operations, although these implementations undergo various changes and revisions that may lead to degradation of existing features and introduction of new features. Cloud based NOSQL will have an issue of consistency but it always guarantees eventual consistency .

III. NO-SQL COLUMNAR ORIENTED DATABESE

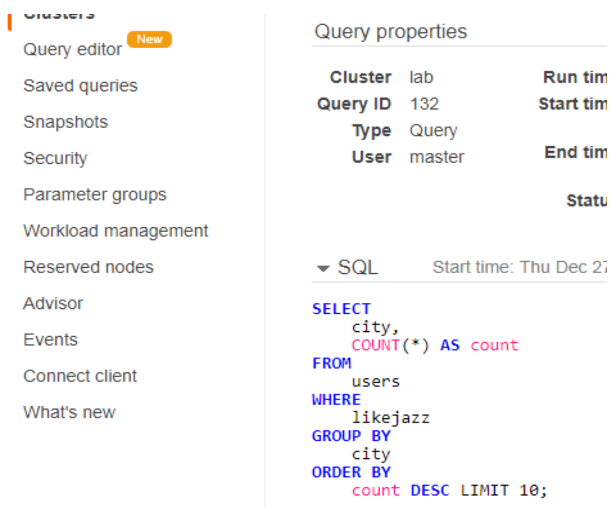
Column-oriented databases (COD) are dealing with the way data is physically stored on disk such that each column is stored continuously in its own separate space/file. This allows for two important things:

a) COD achieve better compression ratio to the order of 10:1 because of ingle data type to deal with at continuous disk space.
b) COD achieve better data read performance because it avoid whole row scans and can just pick and choose the

columns specified in SELECT query.
By using numbering system in columnar databases, algorithms can be used to simplify the retrieval of data. Each Columnar database system has highly sophisticated methods of gaining further performance measures. AWS Redshift ,Azure Hbase in HDinsight, Google Big table are column Oriented NO SQL Data bases.

Nowadays size of data are increasing continuously and also RDBMS makes inefficient use of disk to maintain and store large amount data due to which query processing becomes slow. Thus column-oriented database are more efficient for handling records with the minimum use of disk. Almost all the organization maintains record of data for its further use due to which size of the warehouse is continuously increasing. Organizations are moving towards NOSQL solutions, like key-value oriented DB, Document DB, Columnar Oriented DB and time series DB. Each of these Databases are suitable for specific kind of applications.

In columnar oriented data stores, data are stored and retrieve in columns and hence it can only able to read only the relevant data if required. For accessing large amount of data we can compress data using column-oriented database for efficient storage. For OLAP system row oriented database is not suitable as it takes more time and storage to process data. Below figure 1 is an AWS Redshift Columnar Oriented service .First a data warehouse created which is called Redshift cluster. A table schema is shown in figure 2 for AWS Redshift .Data set with 149970 rows are uploaded on cluster .Later different SQL based queries are fired and performance is recorded. Queries can be fired using AWS Redshift consol or AWS command line interface (CLI). The performance of SQL queries are shown in Table 1. It is found that sum, count, max, min is always efficient in Columnar Oriented Databases because the data is stored vertically as shown in Figure 3 . Considering data compression, faster Read/ Write access, scalability, availability and better performance for aggregate queries, it is advisable to use COD for specific scenario. Also, Columns are stored sequentially. Sequential access is much faster than random access. Column Oriented database can also be compressed more efficiently with similar values stored together. This saves on capacity and values can be read faster. Compression, aggregation queries, scalability and fast to load query are its key benefits. This example demonstrates the table created with single cluster on AWS Redshift with dc2.large node. Figure 3 demonstrates the create Table query and shows the columnar oriented database layout on AWs portal. Create tables query run time recorded is 7.6s .Figure 4 shows group by query performance.



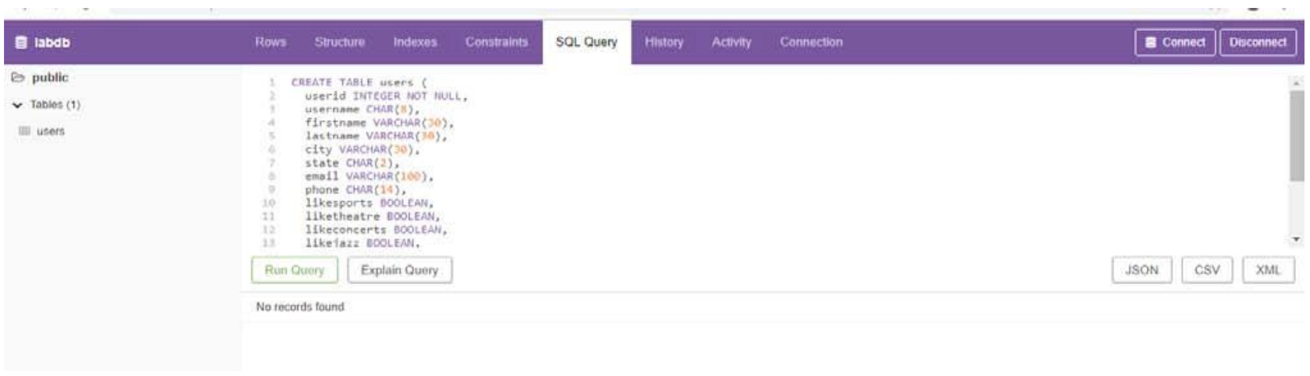
Cluster lab
Query ID 132
Type Query
User master
Run tim
Start tim
End tim
Statu
Start time: Thu Dec 27

```
SELECT
    city,
    COUNT(*) AS count
FROM
    users
WHERE
    likejazz
GROUP BY
    city
ORDER BY
    count DESC LIMIT 10;
```

Figure 4: AWS Redshift with query performance for group by query 7.6s

Id	fn	ln	city	state	email	mobile	liketheater	likesports	Likeconcerts	likejazz	likeclassical	likerock	likebroad	likeProj

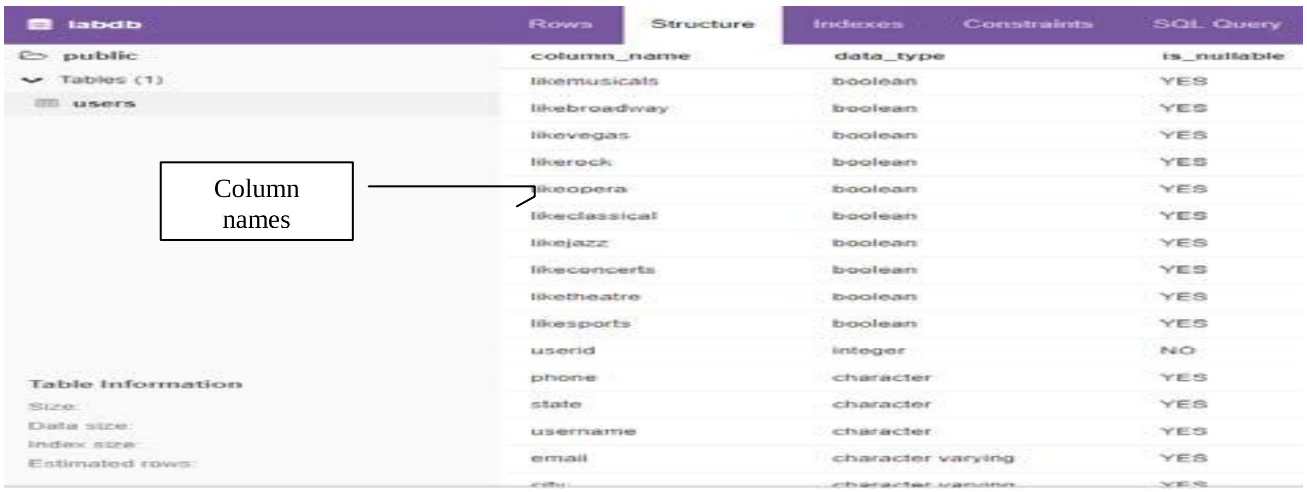
Figure 1: AWS Redshift Table Schema



```
CREATE TABLE users (
  userid INTEGER NOT NULL,
  username CHAR(8),
  firstname VARCHAR(30),
  lastname VARCHAR(30),
  city VARCHAR(30),
  state CHAR(2),
  email VARCHAR(100),
  phone CHAR(14),
  likesports BOOLEAN,
  liketheatre BOOLEAN,
  likeconcerts BOOLEAN,
  likejazz BOOLEAN,
```

No records found

Figure 2: AWS with create table query



column_name	data_type	is_nullable
likemusicals	boolean	YES
likebroadway	boolean	YES
likevegas	boolean	YES
likerock	boolean	YES
likeopera	boolean	YES
likeclassical	boolean	YES
likejazz	boolean	YES
likeconcerts	boolean	YES
liketheatre	boolean	YES
likesports	boolean	YES
userid	integer	NO
phone	character	YES
state	character	YES
username	character	YES
email	character varying	YES
city	character varying	YES

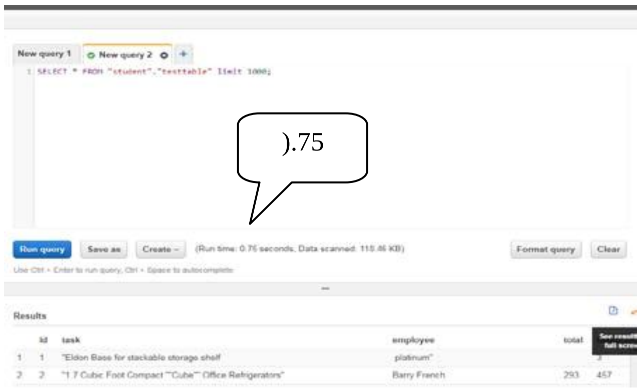
Figure 3: Columnar Oriented DB on AWS Redshift

Table 1: AWS Redshift query performance

Sr.No.	Query	Run time
1	Create table	9.7s
2	Select * from User	5.28s
3	Select userid from User (one Column)	3.93s
4	Select userid,firstname from User (Two Column)	3.98s
5	Select userid,firstname,lastname from User (Three Column)	6.2s
6	Select userid,firstname,lastname,city,state from User (five Column)	5.76s
7	Select count (*) from User	1.85s
8	Select query with group by clause	7.6s
9	Copy from S3	382ms
10	Select count(*) from User limit 100	1ms

IV. ANALYSING BIG DATA

AWS Redshift performance of above table can be enhanced through combination of MPP-Massive Parallel Processing, cluster size and efficient target data compression encoding schemes. AWS Redshift can consider as data warehoused where cluster is provisioned. AWS also provides Athena with standard ANSI SQL which is server less approach, where no cluster provision is required. Data which has to be analyzed can be directly kept in AWS unstructured storage i.e. S3 .It can work on JSON, CSV files uploaded in S3. To use Athena it is required to create DDL statement on Athena consol. In this experiment data which is CSV file with 1000 record is kept in S3. Various quires were fired and performance is s shown in figure 5 .



```
SELECT * FROM "student"."testtable" limit 1000;
```

(Run time: 0.75 seconds, Data scanned: 115.46 KB)

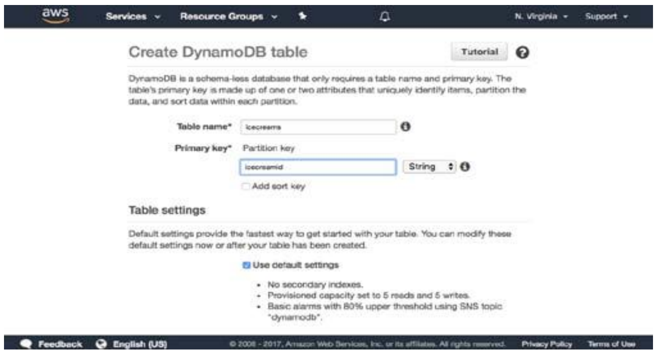
Use Ctrl + Enter to run query, Ctrl + Space to autocomplete

	id	task	employee	total	
1	1	"Eldon Base for stackable storage shelf	platinum"		3
2	2	"1.7 Cubic Foot Compact ""Cube"" Office Refrigerators"	Barry French	293	457

Figure 5: AWS Athena with query run time 0.75s

It is interesting to note that AWS Athena is faster as compared to AWS Redshift in our case . AWS Redshift took 5.25 s for SELECT * from User .Select two columns from table took 1.74 s with AS Athena whereas 3.98 s with

Redshift .initialization time for AWS Athena was significantly less than Redshift because of no cluster initialization in Athena .In this setup for adhoc queries Athena is better choice Scalability is good in Redshift.When dataset is larger , and more queries are be triggered , in that case performance of Redshift will vary depending upon cluster size.AWS also provides Dynamo DB which is key value document and document Data base. As per AWS specification "*DynamoDB can handle more than 10 trillion requests per day and support peaks of more than 20 million requests per second*." Figure 6 shows the DynamoDB from AWS portal.



Create DynamoDB table

DynamoDB is a schema-less database that only requires a table name and primary key. The table's primary key is made up of one or two attributes that uniquely identify items, partition the data, and sort data within each partition.

Table name* icecreams

Primary key* Partition key

icecreamid String

Add sort key

Table settings

Default settings provide the fastest way to get started with your table. You can modify these default settings now or after your table has been created.

✓ Use default settings

- No secondary indexes.
- Provisioned capacity set to 5 reads and 5 writes.
- Basic alarms with 80% upper threshold using SNS topic "dynamodb".

Figure 6: DynamoDB – key value DB on AWS portal

DynamoDB is very well suited for applications likes Catalog ,User profile and content management where schema is not defined or changing for reach item inside a table. Dynamo DB flexible schema less key value architecture uses SSD – solid state drive to store and process the data . It sues Sharding as its load balancing strategy and gives high performance. It is interesting to note that AWS Redshift and Dynamo DB services can be used together. Data arriving from sources can be kept in Dynamo DB, after cleaning it can be transferred to Redshift wherehouse for deep analytics. Dynamo DB pricing model as shown in figure 6 can be on demand or provisioned based on Read/Write request from an application. Reading is reading an item of size 4KB and write request is one write of an item upto 1 KB size. One can also use Auto scaling feature to automate the capacity management for the table.

V. CONCLUSION

Cloud based NOSQL DB has a big role to play in terms of distributed DB, availability, scalability, performance and security. Prominent Cloud providers provide verity of NOSQL DB which differs in architecture and function. A paradigm shift is required from SQL to Cloud based NOSQL as per the desired task. NOSQL databases need to be even more purposefully used than SQL to have efficient operations. NOSQL are designed for large volumes of semi structured or unstructured data. Columnar oriented database like AWS Redshift as a service application are in content management systems, blogging platforms, systems that maintain counters and services that have exponential usage. Bigtable, Cassandra, HBase, Vertica, Accumulo, Hypertable, etc are some examples of column-oriented data store which can be used for specific scenario. Despite the academic and commercial success of NOSQL DB in real time analytics, Big Data, Internet of Things (IoT) , product catalog, Social networking gaming applications, Mobile applications, fraud detection, there are still several interesting directions for future research. In particular, there is a substantial opportunity for hybrid systems .In a given scenario shortcomings of relational database and the strengths of various types of NoSQL databases must be evaluated carefully. We expect that NOSQL data models from cloud will start to find their way into the project undertaken by under graduate and post graduate students and even the main purpose of this paper was to provide the motivation for the same.

REFERENCES

[1] Jagdev Bhogal and Imran Choksi " Handling Big Data using NOSQL" ,IEEE 29th International confrence on Advanced Information Networking and applications workshop,2015
[2] Vandana Bhagat, Arpita Gopal, *"Comparative Study of Row and Column Oriented Database"*, IEEE, 2012, Available at: *https://ieeexplore.ieee.org/document/6495251*
[3] Daniel Abadi, Peter Boncz, Samuel Madden, "The Design and Implementation of Modern Column Oriented Database Systems", IEEE, 2013.
[4] Daniel J. Abadi, Daniel S. Myers, David J. DeWitt, Samuel R. Madden, "Materialization Strategies in a Column-Oriented DBMS", IEEE, 2007.
[5] Suryawanshi Harshal, Rokade Chakrapani, Ambhore Ajay, Rathod Sharad, "Compiler as Service over Cloud", International Journal of Computer Applications,Volume 70– No.1, May 2013.
[6] Uday Chauhan, Mayuri Patole, Yogita Mokashi, SaurabhPadalikar "Online Cloud Based Compilers System", Computer Society Of India ,February 2016.
[7] DBMS Musings, "Apache Arrow vs. Parquet and ORC", 2017, Available at: http://dbmsmusings.blogspot.com/2017/10/apache-arrow- vs-parquet-a nd-orc-do-we.html
[8] TeachMeHANA, "Difference between Row Store and Column Store", May, Available at: http://teachmehana.com/row-store-vs-column-store/
[9] Ansari Mohd. Arshad, Arshiya Khan, Shaikh Sana, Er. Zainab Mirza, "Compilers on Cloud", IJERT, 2013.
[10] Pat Research, "Top 9 Column-Oriented Databases", 2019, Available at https://www.predictiveanalyticstoday.com/top-wide-
[11] columnar-store-d atabases/ Jon Tong, "Demystifying Columnar Databases", April2012, Available at: https://www.youtube.com/watch?v=wYl_YxqTof4
[12] Parallaxinc, "Cloud-Compiler", GitHub, 2015, Available at :https://github.com/parallaxinc/Cloud-Compiler

[13] Jyoti Kharade, Anil Rama Kale, Dhanaji S. Kharade, "NOSQL Database: Opportunities and Applications", International Journal of Computer Sciences and Engineering, Vol.-6, Issue-5, May 2018.

[14] Jared Hilam,"What is a Columnar Database?", Available at https://www.youtube.com/watch?v=8KGVFB3kVHQ

[15] https://docs.aws.amazon.com/redshift/latest/mgmt/welcome.html

[16] https://aws.amazon.com/blogs/aws/new-auto-scaling-for-amazon-dynamodb/

[17] V. S. Dhaka and M. Zainab, "Big Data Management as a Service (BDMaaS) using NOSQL model from Cloud," no. 3, pp. 18–22, 201

[18] V.S.Dhaka and M.zainab "Big Data Analytics as a Service (BDAaaS) System Layer" INDIACom-2017; IEEE Conference

[19] Academind, "SQL vs NoSQL or MySQL vs MongoDB",2018, Available at: https://www.youtube.com/watch?v=ZS_kXvOeQ5Y&t=11

[20] Suryawanshi Harshal, Rokade Chakrapani, Ambhore Ajay, Rathod Sharad, "Compiler as Service over Cloud", International Journal of Computer Applications,Volume 70– No.1, May 2013.

Authors Profile

Dr. Savita Shiwani is PhD in in computer science. She is currently working as a professor and HOD . She is having 15 years of Teaching experience in the area of Computer Science and Engineering. Her area of specialization includes: Artificial intelligence Image processing,Computer Networks, Cloud computing IoT. She has published a good number of research papers in reputed national and international Journals. She has published more than 20 research papers in reputed international journals including Thomson Reuters (SCI & Web of Science) and conferences including IEEE and it's also available online. Her main research work focuses on Artificial intelligenc ,image processing Network Security, Cloud Security and Privacy, Big Data Analytics and Data Mining .

Mrs. Mirza Zainab ,Mtech IT currently working as Assistant professor and HOD. She pursed Bachelor of Computer Engineering in year 2001 and Mtech in the yaer 2006. She is a member of ACM & life member of ISTE . She has published more than 15 research papers in reputed international journals and conferences including IEEE and it's also available online. Her main research work focuses on Cloud computing and Big data.She is awarded with Educational grant from Microsoft for free access of Azure cloud .She has recievd two best paper award at international conference. She is having 16 years of teaching experience.