Database Systems Assignment

NoSQL databases and their comparison against RDBMS.

2012A7PS034P- UTKARSH ASHOK PATHRABE
2012A7PS038P- NITIN LABHISHETTY
2012A7PS090P- GAURAV BANSAL

With the rise of BigData and the need to efficiently handle/use large amounts of data, NoSQL database systems are gaining popularity. What features of these database systems make them better compared to the traditional RDBMS? We compare a NoSQL database, HBase and a SQL database system, Oracle SQL to find out what advantages/ disadvantages do one have over another. Following is comparison of the both based on five parameters crucial to any good database system:

1. Data access time/ performance:

HBase is suitable for querying large amounts of data at very high speeds. For small amounts of data, though the time difference is insignificant, Oracle SQL provides better performance compared to HBase. Oracle SQL is better for smaller datasets due to the query complexity it supports. As the size of data increases and the databases are stressed by running multiple threads processing different queries, HBase provides much faster results compared to Oracle SQL. Selecting records, reading and searching are faster in HBase compared to Oracle SQL probably due to the strict schema followed by SQL database and the additional data stored in order to support complex queries. HBase is light, provides much open method of storing data via document based structure. It is significantly faster for adding new data and accessing existing data hence is the better option in this requirement.

2. Scalability:

Scalability has two sides to it. Upward scalability and outward scalability. Both database systems are equal in terms of outward scalability but in upward scalability, NoSQL database is better. NoSQL database supports Map-reduce operations hence providing it high scalability. Oracle SQL performs well for large amounts of data but requires highly optimized queries to be executed. HBase is designed to handle large amounts of data hence the semantics of querying remain almost the same for varying data sizes. The primary reason for advantage of HBase in this domain is due to the high scalability of the underlying filesystem (HDFS). Hadoop has some features that make it more suitable for large scale data.

3. Physical Storage:

RDBMS storage is very expensive compared to HBase. HBase is significantly superior since it manages to store simple to complex structure definitions as a document hence overlooking the requirement to be concerned with the type of data being stored. Due to the strict schema followed by SQL databases while storing data, there are 'assumptions' made on the type of queries that will be made on the data being stored hence SQL models the data and stores it based on these assumptions. HBase is a distributed, column oriented data storage system hence is light and less expensive in terms of physical storage.

4. Complexity:

Oracle SQL stands out clearly in terms of complexity of queries supported. Oracle SQL allows us to write complex, nested, high level queries with ease. HBase on the other hand though does not provide great functionalities for writing complex queries, at some level it allows user to write more complex queries than Oracle SQL. The user has to manually write and create multiple queries to get

the desired result but at the end, it enables the user to write any sort of query imaginable. Overall, complex queries are supported by both but the ease of writing them is more on Oracle SQL.
5. Data availability and Integrity:

CAP theorem is used for assessing data availability and integrity of databases. CAP stands for Consistency, Availability and Partition tolerance. A shared database system can have at most two of these three features. Oracle SQL follows ACID principle (Atomicity, Consistency, Isolation and Durability) while HBase follows BASE principle (Basically Available, Soft state, Eventually consistent). Oracle SQL (or systems following ACID principles) have strict rules on consistency compounded by the strict schema design to be followed. This is trade-off for data availability and performance. While data availability is high in HBase, due to their schema free nature, data integrity is not much of a concern. Rows use only the columns they require which enables easy adding of columns, column families by just using the namespace of the tables. But boosting performance and integrity does affect data consistency which is one of the drawbacks of HBase (NoSQL database systems).

Overall SQL (Oracle SQL in this case) systems provide advantages in Complexity of queries, Data consistency, maintaining standards and theory based models. But NoSQL (HBase used in this comparison) systems are superior in **performance**, **data access**, **scalability**, **physical storage**, **support of map reduce and complex data types**. Hence NoSQL systems are better for handling the upcoming needs of handling large, raw unprocessed data. Their drawback though is unavailability of standards, strong development environment providing active user help and support and consistency for secure database purposes. Though they cannot serve as a replacement for the current RDBMS, they are advantageous in certain aspects and can certainly be used in applications of database management systems in big data environments.

Following are some of the queries we have written to create the following schema to test some of the above mentioned properties in HBase:

1. HBase queries written in HBase shell:

```
> create
'employee',{NAME=>'basic'},{NAME=>'history',VERSIONS=>5},{NAME=>'education'},{NA
ME=>'address'}
> put 'employee','employee1','basic:empid','101'
> put 'employee','employee1','basic:name','Utkarsh'
> import java.util.date
> put 'employee','employee1','basic:dob',new Date(1994, 9,10).toString()
//DATE ABOVE IS 10 October 1994
> put 'employee','employee1','basic:photo','Yes'
> put 'employee','employee1','basic:startdate',new Date().toString()
> put 'employee','employee1','history:title','Analyst'
> put 'employee','employee1','history:title','Programmer'
> put 'employee','employee1','education:high_school','Science'
> put 'employee','employee1','education:high_bachelor','Computers'
> put 'employee','employee1','address:door','260'
> put 'employee','employee1','address:street','Gandhi'
> put 'employee','employee1','address:city','Pilani'
> put 'employee','employee1','address:state','Rajasthan'
```

Similar commands for other employees can also be generated.

NOTE: To execute any alter statements first disable table using the command *disable '<table-name>'* and then enable it after making the required changes using the command *enable '<table-name>'*

```
//Altering table for maxfile size of 500KB
> disable 'employee'
> alter 'employee', METHOD=>'table_att',MAX_FILESIZE=>'512000'
> enable 'employee'
```

To retrieve data according to:-

```
1) table name        -     scan '<table-name>'
                           scan 'employee'
2) row ID            -     get '<table-name>','<reference>'
                           get 'employee','employee1'
3) column family  -     get '<table-name>','<reference>',
                  {COLUMN=>'<column-family>'}
                           get 'employee','employee1',
                        {COLUMN=>'basic'}

                           get '<table-name>','<reference>',
                  {COLUMN=>'<column-family>', VERSIONS=>'<n>'}
                           get 'employee','employee1',
                  {COLUMN=>'history',VERSIONS=>2}
4) key value         -     get '<table-name>','<reference>','<key>'
                           get 'employee','employee1','basic:name'
```

To update any value just put the new value with the existing key
for instance if you want to update the key 'name' for 'employee1' just execute the command

```
> put 'employee','employee1','basic:name','<NEW_NAME>'
```