Implementation of an Efficient MongoDB NoSQL Explorer for Big Data Visualization

Chaitanya.P[#1], Ranjan H.P[#2], Kiran T.S [#3] Anitha.K*[4]
UGstudents, DeptofCSE, RRCE, Bangalore, Karnataka, India[1,2,3]
Associate Prof. Rajarajeswari College of Engineering, Bangalore, India[4]

ABSTRACT- With the emergence of Big Data, the use of NoSQL (Not only SQL) has increased among internet companies and other enterprises. Benefits include horizontal scaling, finer control over availability and simplicity of design. NoSQL databases are considered as an alternative to relational databases, as its schema less data model is considered to be better for handling the large volumes of structured and unstructured data. This paper aims to introduce the concepts behind NoSQL and provide arguments for and against adopting NoSQL. A small library application has been developed to assess the stated benefits of NoSQL and NOSQL and compare its performance with the existing systems like MySQL/ Oracle and a re-usable tool called Mongo-sight is developed to visualize the data stored in it.

Keywords— NOSQL, SQL, Databases, structured data, unstructured data, Big Data, Mongo-Sight, MongoDB.

I. INTRODUCTION

Big data is a buzzword, catch-phrase, which means massive volume of both structured and unstructured data which is so large is difficult to process using traditional database and software techniques. It is estimated that volume of data is increasing 40% per year, and will grow 44 times between 2009 and 2020. Most of the data is unstructured as it is in the textual form. For example NoSQL databases are mostly used to collect and store data related to social media.

II. OVERVIEW OF NOSQL

NoSQL databases are non-relational and can accommodate unstructured data. It is not a replacement for SQL database but rather compliments it, both these technologies can coexist.

One of the key differences is that SQL databases have rigid schemas while NoSQL offeres a flexible schema that can be altered without service disruption.

There are four categories of NoSQL databases:

A key-value store, or key-value database, is a data storage paradigm designed for storing, retrieving, and managing associative arrays, a data structure more commonly known today as a dictionary or hash. Dictionaries contain a collection of records, objects which in turn have many different fields within them, each containing data. These records are retrieved & stored using a key that uniquely identifies the record, and is used to quickly find the data within the database.

Key-value stores work in a very different fashion from the better known relational databases (RDB). RDBs pre-define the data structure in the database as a series of tables containing fields with well defined data types. Exposing the data types to the database program allows it to apply a number of optimizations. In contrast, key-value systems treat the data as a single opaque collection which may have different fields for every record. This offers considerable flexibility and more closely follows modern concepts like object-oriented programming. Because optional values are not represented by placeholders as in most RDBs, key-value stores often use far less memory to store the same database, which can lead to large performance gains in certain workloads. Figure 1 shows an example of a Key Value Database.

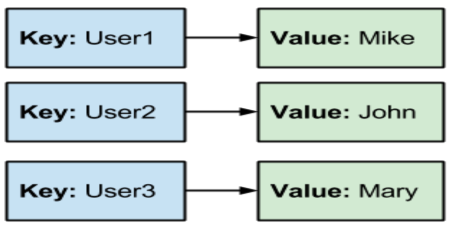
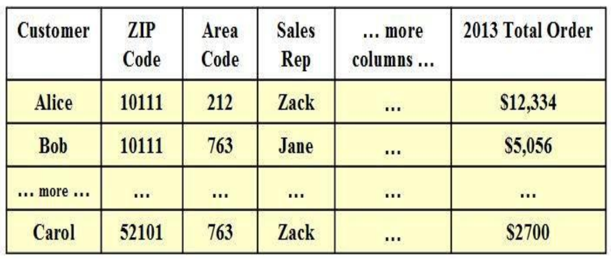


Figure 1:Key Value Database

A column-oriented DBMS is a database management system (DBMS) that stores data tables as sections of columns of data rather than as rows of data. In comparison, most relational DBMSs store data in rows. This column-oriented DBMS has advantages for data warehouses, clinical data analysis, customer relationship management (CRM) systems, and library card catalogs, and other ad hoc inquiry systems where aggregates are computed over large numbers of similar data items. An example of a column Oriented database using Customer Information is shown in Figure 2.



Customer	ZIP Code	Area Code	Sales Rep	… more columns …	2013 Total Order
Alice	10111	212	Zack	…	$12,334
Bob	10111	763	Jane	…	$5,056
… more …	…	…	…	…	…
Carol	52101	763	Zack	…	$2700

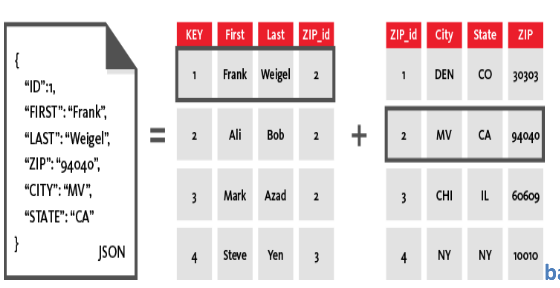
Figure 2:Column Oriented Database

Document-oriented databases are inherently a subclass of the key-value store, another NoSQL database concept. The difference lies in the way the data is processed; in a key-value store the data is considered to be inherently opaque to the database, whereas a document-oriented system relies on internal structure in the document order to extract metadata that the database engine uses for further optimization. Although the difference is often moot due to tools in the systems, conceptually the document-store is designed to offer a richer experience with modern programming techniques. XML databases are a specific subclass of document-oriented databases that are optimized to extract their metadata from XML documents. Graph databases are similar, but add another layer, the relationship, which allows them to link documents for rapid traversal.

Document databases contrast strongly with the traditional relational database (RDB). Relational databases are strongly typed during database creation, and store repeated data in separate tables that are defined by the programmer. In an RDB, every instance of data has the same format as every other, and changing that format is generally difficult. Document databases get their type information from the data itself, normally store all related information together, and allow every instance of data to be different from any other. This makes them more flexible in dealing with change and optional values, maps more easily into program objects, and often reduces database size. This makes them attractive for programming modern web applications, which are subject to continual change in place, and speed of deployment is an important issue.

graph database is a database that uses graph structures for semantic queries with nodes, edges and properties to represent and store data.

Most graph databases are NoSQL in nature and store their data in a key-value store or document-oriented database. In general terms, they can be considered to be key-value databases with the additional relationship concept added. Relationships allow the values in the store to be related to each other in a free form way, as opposed to traditional relational databases where the relationships are defined within the data itself. These relationships allow complex hierarchies to be quickly traversed, addressing one of the more common performance problems found in traditional key-value stores. Most graph databases also add the concept of tags or properties, which are essentially relationships lacking a pointer to another document. data



```
{
  "ID":1,
  "FIRST":"Frank",
  "LAST":"Weigel",
  "ZIP":"94040",
  "CITY":"MV",
  "STATE":"CA"
}
```

KEY	First	Last	ZIP_id
1	Frank	Weigel	2
2	Ali	Bob	2
3	Mark	Azad	2
4	Steve	Yen	3

ZIP_id	City	State	ZIP
1	DEN	CO	30303
2	MV	CA	94040
3	CHI	IL	60609
4	NY	NY	10010

base

Figure 3:Document-oriented

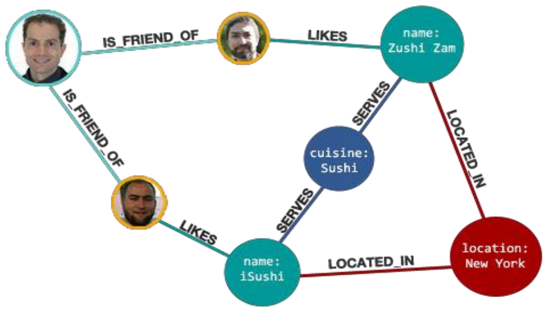


Figure 4: Graph database

III. NoSQL PROTOTYPE

A prototype system of a library inventory will be developed which will act as a platform to compare both SQL and NoSQL databases. We examine Big Data in the context of libraries. Libraries not only store huge quantities of electronic data for their product and membership information. Present-day libraries also capture Data from social media sites such as Facebook, Twitter etc. whereby library members can give feedback and ratings regarding the quality of the products and lending service.

RELATIONAL DATABASE DESIGN – SQL

A relational database organizes data in *tables* (or *relations*). A table is made up of rows and columns. A row is also called a *record* (or *tuple*). A column is also called a *field* (or *attribute*). A database table is similar to a spreadsheet. However, the relationships that can be created among the tables enable a relational database to efficiently store huge amount of data, and effectively retrieve selected data. Libraries relational database schema is designed by ER modelling. The library database has 4 many-to-many relationships, and 2 one-to-many relationships. All the tables have been fully Normalized. This process removes all redundant data from tables as it improves data integrity as well as storage efficiency. As additional table joins may be required during data retrieval therefore Normalization can impact performance.

The libraries database employs multiple table inheritance to store common attributes in a generic Product table (see Figure 5).

There are various relational databases that store data in tables and operate through queries. Oracle Application Express (APEX) was selected for implementing the libraries relational database. Oracle APEX is a freeware software development environment running inside the Oracle database.

Its web browser interface permits inexperienced programmers to quickly develop data orientated applications. APEX uses wizards and declarative programming to form powerful data entry applications, and contains a graphical query builder. Its SQL Workshop component is employed to manage database objects and run ad hoc queries.

There are several NoSQL databases that store data in JSON like documents and provide atomic scan write operations. When deciding whether or not to use NoSQL, companies need to consider the business model, ACID transactions demand, cost and other requirements. MongoDB was chosen for implementing the libraries document information,

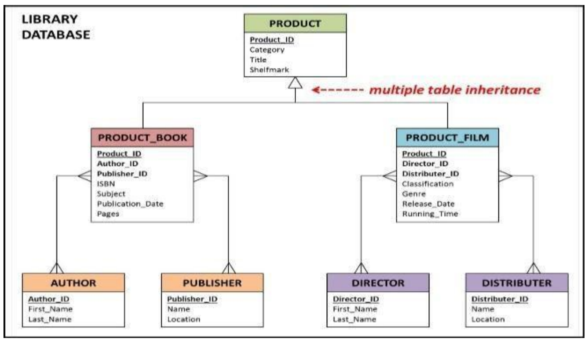


Figure 5: *Document database design - NOSQL*

Document databases store data in the form of documents that are JSON like key value pairs which are similar to rows in relational databases. Documents contain many different key value pairs, and keys may hold other key values. Documents are stored in collections; these are groups of related documents that have shared common indexes and are similar to tables in relational databases.

Document databases have a flexible schema. SQL databases requires the tables schema to be defined before inserting the data, but collections do not impose document structure. This flexibility enables the documents to match data fields of any entity, even when the data is varied substantially.

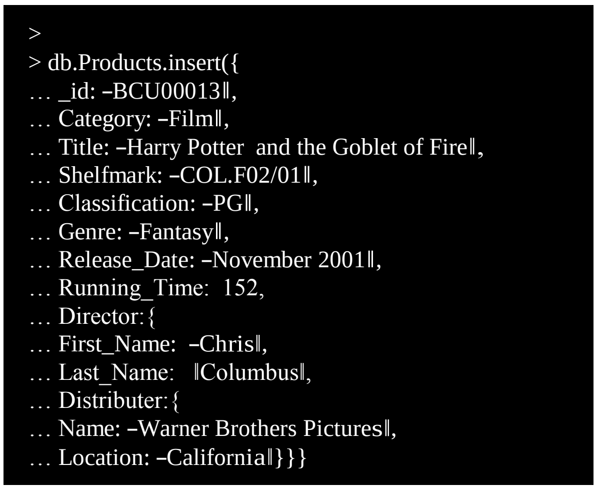
Document databases can be represented using either the reference or embedded approaches. Related Data in Embedded documents is stored in a single document. This denormalized data model allows systems to query and update related data in a single database operation.

because its flexible data model permits a prototype system to evolve during development while not modifying existing data. Its embedding model eliminates join operations and enhances query performance. it's also horizontally scalable and can reduce work by adding additional commodity servers or cloud instances. MongoDB provides a information system that stores wealthy data structures, executes complicated queries and scales out. MongoDB permits the library to insert data without a predefined schema and this is often useful for storing products with totally different attributes.

IV. IMPLEMENTATION

Firstly a relational database solution was constructed using Oracle APEX. thenceforth a document info solution was constructed using MongoDB; this style incorporates both embedding and reference data models. the aim of this section is to Illustrate the fundamental steps of each Applications have been developed for both Android devices implementation and highlight the significant variations between both approaches. MongoDB employs insert operations to create data; the db.collection.insert() statement adds new documents to a set. The aid field is automatically generated for a new document if the field isn't specified. The libraries document info system employs both embedded documents and references; this enables the author to demonstrate both data models.

implementation and highlight the significant variations between both approaches. MongoDB employs insert operations to create data; the db.collection.insert() statement adds new documents to a set. The aid field is automatically generated for a new document if the field isn't specified. The libraries document info system employs both embedded documents and references; this enables the author to demonstrate both data models.

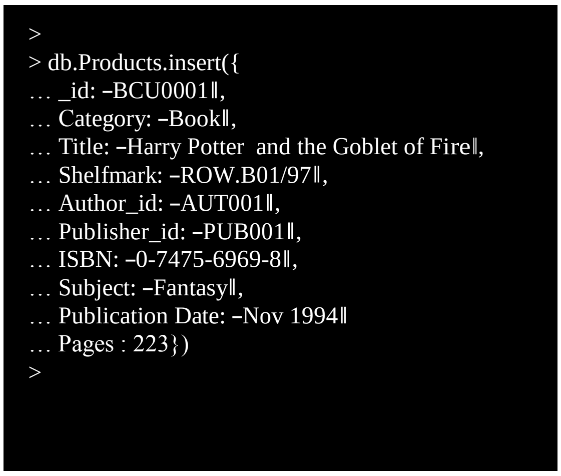


```
>
> db.Products.insert({
… _id: –BCU00013‖,
… Category: –Film‖,
… Title: –Harry Potter  and the Goblet of Fire‖,
… Shelfmark: –COL.F02/01‖,
… Classification: –PG‖,
… Genre: –Fantasy‖,
… Release_Date: –November 2001‖,
… Running_Time:  152,
… Director:{
… First_Name:  –Chris‖,
… Last_Name:   ‖Columbus‖,
… Distributer:{
… Name: –Warner Brothers Pictures‖,
… Location: –California‖}}}
```

Figure 6a: Insert document into products collections

A. EMBEDDED DATA MODEL

Embedded documents store connected data in a single document; film product documents embed the director and distributor attributes. the subsequent statement inserts a movie into the products collection (see Figure 6b):



```
>
> db.Products.insert({
… _id: –BCU0001‖,
… Category: –Book‖,
… Title: –Harry Potter  and the Goblet of Fire‖,
… Shelfmark: –ROW.B01/97‖,
… Author_id: –AUT001‖,
… Publisher_id: –PUB001‖,
… ISBN: –0-7475-6969-8‖,
… Subject: –Fantasy‖,
… Publication Date: –Nov 1994‖
… Pages : 223})
>
```

V. CONCLUSION AND FUTURE WORK

Both prototype systems effectively address the libraries data storage needs, however dissent in their approach. APEX requires table structure to be defined before adding data whereas MongoDB doesn't explicitly create collections; document structure is defined automatically during the primary

data insert. MongoDB allows the library to either embed product attributes into one document or reference the data in another document. but with a normalized SQL database, product attributes may well be spread over various tables; therefore, complicated SQL joins are needed, to view all the attributes of a product. also referential integrity rules specify that the libraries SQL database should solely insert records with Author_ID into the Product_Book table, if the Author_ID exists within the Author table. The NoSQL database is very efficient and permits the library to obtain all film
attributes using one simple query. NoSQL databases make it possible to realize great

value from big data and empowers businesses to be additionally agile and scalable; this helps businesses accomplish their strategic goals and generate new revenue streams. Most new data is unstructured and the rigid schema based approach adopted by relational databases, makes it impossible to include all new data types. sensor data may be used for exploring the concept of data variety further. Future work on using NoSQL to evaluate the remaining three V's of big data is needed. Benchmark tests may be conducted to assess the handling of data volume and velocity with the deployment of NoSQL on Hadoop clusters and also the use of real-time social media data. The social media data may be in the form of text-based product reviews and also the prototype application could be extended to include sentiment analysis. NoSQL databases are becoming a worthy alternative to relational databases, as its dynamic data model is far better for managing large quantities of unstructured data; Additionally, its schema may be modified without downtime or servicedisruption.