IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 3 Issue 6, June 2016
ISSN (Online) 2348 – 7968 | Impact Factor (2015) - 4.332
www.ijiset.com

VISUALIZING CLASS DIAGRAM USING OrientDB DATA-STORE

Sawinder Kaur
CSED, Thapar Univesity
Patiala (Punjab), India
sawinderkaurvohra@gmail.com

Karamjit Kaur
CSED, Thapar University
Patiala (Punjab), India
karamjit.kaur@thapar.edu

Abstract— Relational databases are providing storage for several decades now. The term NoSQL broadly covers all non-relational databases that provide scalable and schema-less model. NoSQL databases are used by major organizations operating in the era of Web 2.0. Different categories of NoSQL databases are key-value pair, document, column-oriented and graph databases which enable programmers to visualize the data closer to the format used in their application. In this paper, class diagram has been merged with OrientDB through Java API to visualize the class diagram as OrientDB graph. OrientDB is the only database which supports both graph and document database, also provides support for both inheritance and polymorphism.

Keywords- OrientDB, NoSQL, Class Diagram.

I. INTRODUCTION

NoSQL is a next generation [6] database which addresses these properties: non-relational, non-ACID[12], distributed, open-source and horizontally scalable , schema-less**.** The original intention has been modern web-scale databases. Rise of NoSQL [15] databases are challenging the dominance of relational databases (dominated the software industry for longer period). There is an impedance mismatch between the relational data structures and the in-memory data structures. A NoSQL database allows developers to develop without having to convert in-memory structures to relational structures.

Key-value is the simple NoSQL data stores to use from an API perspective. User can get the value for a key, put the value for a key or delete a key from the data store. The value is just stored without knowing what is stored inside. Key-value stores always use primary-key and use a hash table where a pointer points to a particular item of data. Key-based lookups results in lesser query execution time since values can be anything like objects, hashes etc. resulting in flexible and schema-less model appropriate for today's unstructured data, hence gives a great performance and can be easily scaled. Key-value databases are Riak(Basho) [8], Redis(VMware) [8], Amazon DynamoDB [5], and Couchbase [15]. To update part of a value or query the database, this method is not ideal.

Documents are the prime concept in document databases. The database stores and retrieves documents which can be XML, JSON, BSON etc. These documents are self-describing and form hierarchical tree data structures which can consist of maps, collections, and scalar values. For example one could search for all documents in which "City" is "Patiala" that would deliver a result set containing all documents connected with any "3 Storey Office" that is in that particular city. Apache CouchDB[9] and MongoDB [15] are popular examples of a document store. CouchDB uses JSON to store data, JavaScripts its query language using MapReduce and HTTP for an API. MongoDB is designed to be able to face new challenges such as horizontal scalability, high-availability and flexibility to handle semi-structured data. MongoDB has typical applications in content management systems, mobiles, gaming and archiving. Document style databases are schema-less so it makes addition of fields easy to JSON without defining changes first.

In Column-oriented/ Wide-table data stores, data is stored in cells grouped in columns of data rather than as rows of data. Columns are logically grouped into column families. Column families hold inside a virtually unlimited number of columns that can be created at runtime or the definition of the schema. Read and write is performed using columns rather than rows. In comparison, most relational DBMS reserve data in rows, the benefit of storing data in columns is fast search, access and data aggregation [7]. Relational databases store a single row as a continuous disk entry. Different rows are stored in different regions on disk whereas Columnar databases store all the cells corresponding to a column as a continuous disk entry making the search and access faster. For example: To query the titles from a bunch of a million articles will be a heedful task while using relational databases as it will go over each location to get item titles. On other side, with just one disk access, title of all the items can be attained. Popular open source column-oriented databases are Hypertable [8], HBase [15] and Cassandra [15]. Hypertable and HBase are derivatives of BigTable where as Cassandra takes its features from both BigTable and Dynamo.

In Graph NoSQL Database, there is no rigid format of SQL, tables and columns representation, a flexible graphical design is instead used which is perfect to address scalability concerns. It does not require a pre-defined schema which leads to easier adaptation to schema evolution. It allows to store entities and relationships amid these entities. Relations are known as edges that can have properties. Edges [9] have directional significance and nodes are organized by relationships which allow you to find interesting patterns among the nodes. The graphs can be correlated and interpreted in different ways. Mostly, when we store a graph-like structure in RDBMS, it is for a single type of relationship, adding another relationship to the mix usually means a lot of schema changes and data movement which is easily done when graph database is used. In relational databases graph is modelled beforehand based on the Traversal one wants; if the Traversal changes, the data will be changed whereas in graph databases, traversing the joins or relationships is very fast [13]. The relationship between nodes is not calculated at query time but is actually prevailed as a relationship. Traversing persisted relationships is faster than calculating them for every query. Example: Social networking websites where relationships among data are as important as data itself are best candidates for graph-based storage. More than 20 graph databases are available of which few are proprietary and others open-source, popular ones are Neo4j [8], Titan [15], OrientDB [8], AllegroGraph [15], InfiniteGraph [15] etc.

OrientDB(Document+Graph) - OrientDB is a 2nd Generation Distributed Graph Database [10] and the first Multi-Model Open Source NoSQL DBMS that carries together the power of graphs and the flexibility of documents into one scalable high-performance [13] operational database. First generation Graph Databases lack the features that Big Data demands: multi-master replication[6], sharding[6] and more flexibility for modern complex use cases. OrientDB is incredibly fast as can store 220,000 records per second [10] on common hardware. Even for a Document based database, the relationships are handled as in Graph Databases with direct connections within records. You can traverse parts of or entire trees and graphs of records within few milliseconds. OrientDB supports schema-less, schema-full and schema-mixed modes [10], has a strong security profiling system based on roles and users, supports **SQL** amongst the query languages. Being a document database one can store any document on a vertex, being a graph database one can introduce new edges and properties. It allows schemas to be introduced at runtime.

Class diagram with OrientDB helps to store the state of the system. Almost for every system a domain model (or logical information model) is framed which describes what information the system must maintain. The state of the art for shaping such models is to build an object-oriented class diagram, typically in UML. This model represents classes with their properties and associations among classes. With most databases there is some impedance mismatch when mapping the canonical model as in Relational model there is no support for inheritance, polymorphism and relationships have to be mapped into keys. In graph databases there is also no support for polymorphism, inheritance and complex properties introduces new vertices. A document database also does not provide support for polymorphism and provides very limited support for relationships. In OrientDB the mapping eliminates all impedance mismatch as object becomes a vertex, complex properties are handled by documents, provides explicit support for relationships, inheritance and polymorphism.

II. MOTIVATION

To create a graph that contains a class diagram using the Java API OrientDB so that it becomes easy to analyse how the class diagram is depicted in graph database. It helps to easily understand how the classes are related to each other, how polymorphism is used. OrientDB graph is easily extensible and this can be achieved by adding information about methods or by purporting the meta-data. In this manner information can be managed about annotated methods and released revisions.

Projects need some extent of run-time configuration where user could configure the rules of the data structures stored and hence complexity increases. If a relational database is used behind any application then high complexity is seen between joins to retrieve the data. A schema-free document database could simplify complexity problem. OrientDB allows schemas to be introduced at runtime and provide record level security which means any record or class can be altered to extend any other - including vertices and edges in the graph.

To measure the performance based on response time of the retrieved data using JAVA API OrientDB.

III. NoSQL DATA-STORES

NoSQL referred to (not only SQL) or (non relational [3]) database which provides a mechanism for storage and retrieval of data modelled different from the tabular relations used in relational databases. They have existed since the late 1960s, but did not obtain the NoSQL signature until its popularity in the early twenty-first century when Web 2.0 companies such as Facebook, Google and Amazon.com [15]. NoSQL databases are widely used in Bigdata, real-time web [2] applications and also supports SQL like query languages.

A. DOCUMENT-ORIENTED DATA MODEL OR DOCUMENT STORE

Is designed for storing, retrieving, and managing document-oriented information also called as semi-structured data [3]. The popularity of the term document-oriented database has grown [1] with the help of the term NoSQL itself. XML databases [4] are a subclass of document-oriented databases. Document-oriented databases are intrinsically a subclass of the key-value store [4], another NoSQL database concept. Example the following document is encoded in JSON.

```
{
 First_Name:”Saw”,
 Last_Name:”Kaur”,
  Hobby:”Swimming”
}
```

Data processing	In a key-value store, the data is contemplated to be inherently opaque to the database.
	In document-oriented system relies on internal structure in order to extract metadata.
Data storing	Relational databases [11] store data in separate tables defined by the programmer where a single object may spread across various tables.
	Document databases store every information for a given object in a single instance.

TABLE 1: Comparison of document-oriented and relational database.

B. KEY-VALUE STORE OR DATABASE

Is designed for storing, retrieving, and managing associative arrays, data structure. Dictionaries (data structures) contain a collection of objects or records, which have many different fields within them, each containing data. These records are stored and fetched using a key that uniquely distinguishes the record, and is used to quickly find the data within the database.

KEY	VALUE
K1	XXX,ZZ
K2	SSS,5647,8
K3	BBB,10/04/2016
K4	AAA,CCC
K5	AAAA,BBBB

Fig. 1.Key-Valued Database

Data Structure	RDB pre-defines the data structure [11] in the database as a series of tables containing fields with well defined data types.
	Key-value systems treat data as a single opaque collection which have distinct fields for every record.
Memory	Optional values are represented by placeholders in RDB
	Key-value stores often use far less memory as optimal values are not represented in it.

TABLE 2:Comparison of key-valued and relational database

C. GRAPH-BASED DATA MODEL

It uses graph structures for semantic queries with nodes, edges and properties are used to represent and store the data. A key concept of the system is the graph which directly relates data items in the store.

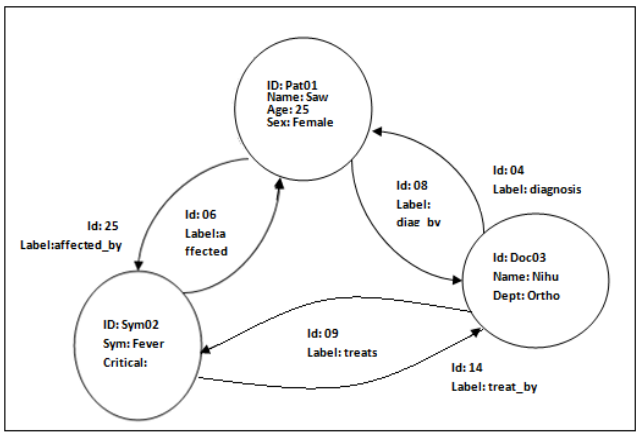

Fig. 2.Graph Database

D. COLUMN-BASED DATA MODEL

A column of a distributed data store is a NoSQL object of the lowest level in a key space. It is a tuple which has three elements [6] Unique name to refer the column, Value to tell the content of column of different types, Timestamp to determine the valid content. Example- In JSON notation three columns are given in Fig3.

```
{
Street:{
name:"sa",value:"16",timestamp:167453
        },
Zip:{
name:"zp",value:"111",timestamp:104111
        },
}
```

Fig. 3.Column Database

E. DOCUMENT AND GRAPH DATABASE

OrientDB can be used in multiple ways as it can be used as document database, graph database giving competition to MongoDB [15] and Neo4J [8][15]and also can be used as an Object-Oriented Database. For better performance Database integration is required which implies the integration [17] and aggregation of data from dif databases within or outside the organization and using the integrated data in many real time applications. New technology like cloud computing[14], Bigdata [2] came into existence, there is a need of sharing the resources and need to achieve consistency also. There are different platforms, different query languages, different data models[16], different dependencies among databases and applications so integration has become a capsule to solve this problem.

IV. OrientDB

OrientDB [10] as a document database can store documents. OrientDB can take an arbitrary document, a JSON document and can store it. After it has been stored one can query it using path expressions. If only document databases are used then it does not supports inheritance and provides no support for relationships.

OrientDB as a graph database which implement the relationships as first class citizens called edges and edges connect vertices. A vertex, in graph databases, is a simple cluster of name-value pairs. In OrientDB each document in document database acts as a vertex in graph database. If only graph databases are used then it does not supports inheritance, polymorphism. Complex properties introduces new vertices even if not required, hence no individuality. In OrientDB, mapping the two databases eliminates all impedance mismatch.

V. CLASS DIAGRAM

It represents a movie recommendation system where one can login and rate the movies according to their aspects. Users can add their favourite movie to the wishlist and can remove from the wishlist. A new user has to create their account and email verification will be done by the system. The type of movie: action, fiction, love story etc. one selects, similar type recommendations will be shown. When a movie is selected, its name, its year and image will be shown when data is retrieved from knowledge base. Based on the selected movie, its aggregated rating will also be shown.

IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 3 Issue 6, June 2016
ISSN (Online) 2348 – 7968 | Impact Factor (2015) - 4.332
www.ijiset.com

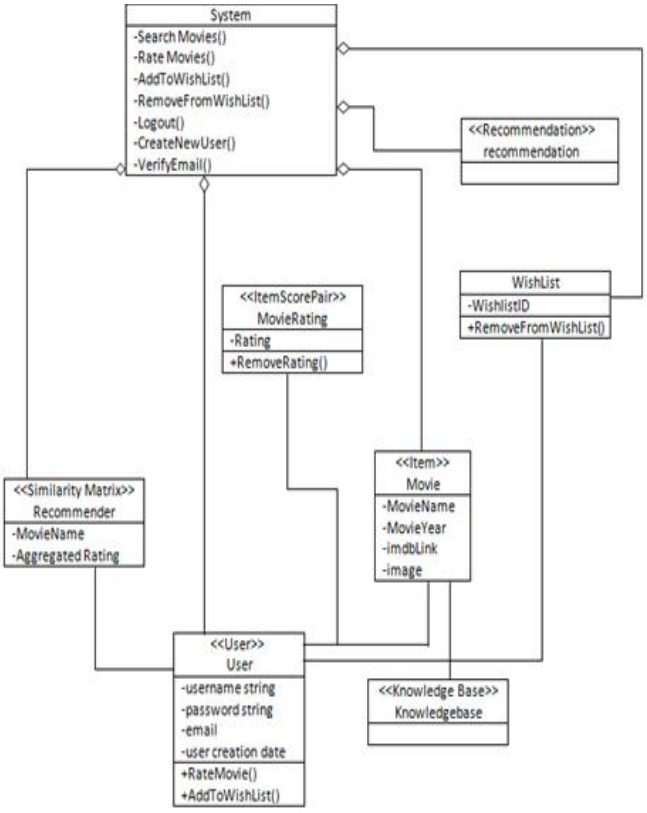

Fig. 4. Class diagram of case study

V1. ANALYSING CLASS DIAGRAM USING ORIENTDB

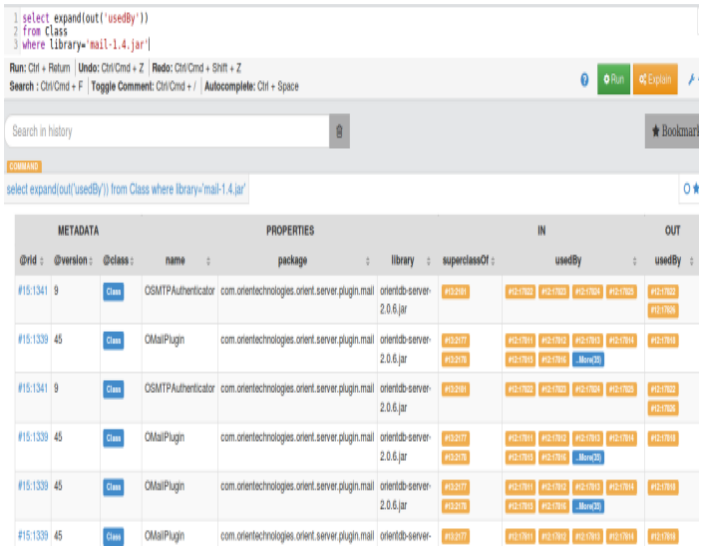

Fig. 7. Time required to retrieve and update the data

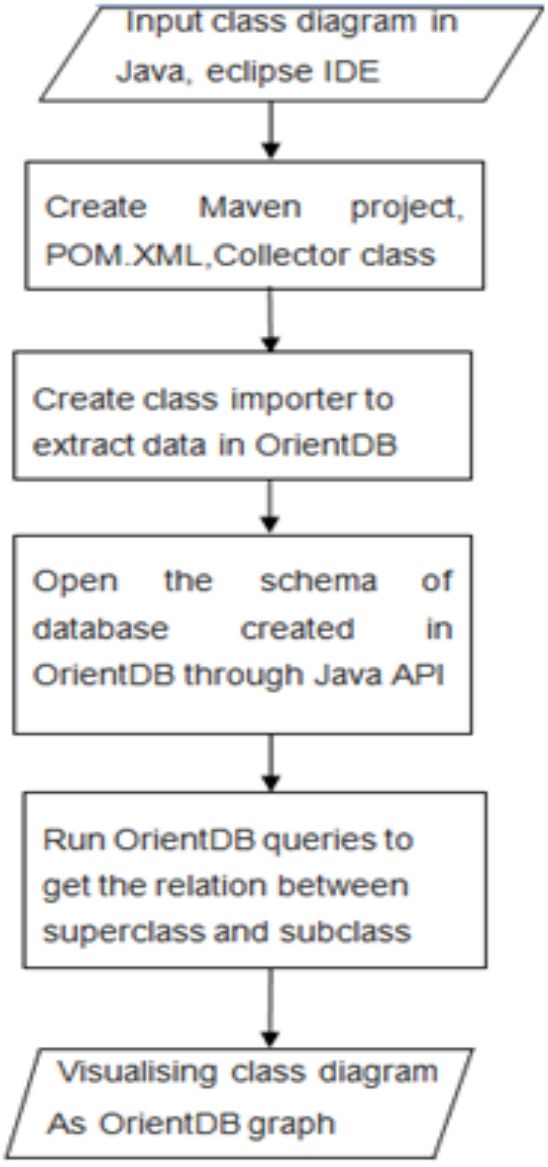

Fig. 5. Flowchart to represent class diagram in OrientDB

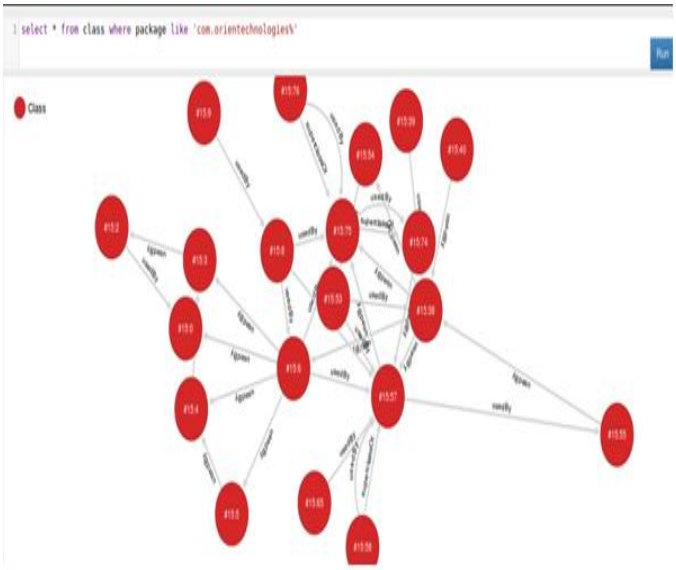

Fig. 6. Class Diagram in form of Graph

VII. PERFORMANCE ANALYSIS

RAM	2GB
HDD	40 GB
Internet connection	DSL
Processor	Intel Pentium Processor 4, 2.4 GHz

TABLE 3: Hardware Environment

Operating System	Windows 7, Ubuntu 14.04
Database	OrientDB 2.0.6
Programming Language	Java
Technologies	Eclipse

TABLE 4: Software Environment

VIII. CONCLUSION

NoSQL is a complementary product for handling issues of scalability, complexity and performance. Non-relational databases provide many enhancements over traditional relational databases [11] such as increased scaling across commodity servers or cloud instances, non-adherence to rigid schema for inserting data and hence ease in capturing of different type data without many changes at schema level.

In this paper, we discussed about NoSQL databases and OrientDB which together provides better features by using both document and graph databases. Class diagram is very popular among application developers, but the concept together with non-relational databases is yet to come. To the best of our knowledge, there is no publication that explained class diagram using OrientDB and querying in it to retrieve and update the class diagram without affecting the other classes. Due to limit on length, only two classes of NoSQL Databases: Document-oriented and Graph-based databases together have been covered in this paper. A case-study have been explained and considered to illustrate the way of Class diagram. With the help of five queries data relationship between classes have been depicted through graph in OrientDB and their performance have been noted down.

REFERENCES

[1]DB-Engines Ranking per database model category [Online]. Available: http://dbengines. com/en/ranking_categories

[2]C Snijders, U Matza, UD Reips (2012). "'Big Data: Big gaps of knowledge in the field of Internet". International Journal of Internet Science 7: 1–5

[3]Leavitt, Neal. "Will NoSQL Databases Live Up to Their Promise?".IEEE, 2010

[4]Grolinger, K. Higashino, W. A. Tiwari, A. Capretz, M. A. M. (2013). "Data management in cloud environments: NoSQL and NewSQL data stores" Springer, 2014.

[5]"Amazon helped start the "NoSQL" movement [Online]. Available: http://www.wired.com/2012/01/amazon-dynamodb/

[6]NoSQL DEFINITION: Next Generation Databases mostly addressing some of the points: being non-relational, distributed, open-source and horizontally scalable[Online]. Available: http://nosql-database.org/

[7]Chaker Nakhli. "Cassandra's data model cheat sheet: Data model elements: Column"[Online]. Available:https://en.wikipedia.org/wiki/Column_(data_store)

[8]O Hajoui, R Dehbi, M Talea - Journal of Theoretical 2015 – "Advanced comparative study of the most promising NoSQL and NewSQL databases with a multi-criteria analysis method "

[9]Renzo Angles and Claudio Gutierrez. "Survey of graph database models". ACM Computing Surveys (CSUR), 40(1):1, 2008.

[10]Claudio Tesoriero. "Getting Started with OrientDB". Packt Publishing Ltd, 2013.

[11]RDBMS dominate the database market, but NoSQL systems are catching up". [Online]DB-Engines.com, 2013

[12]Bogdan Tudorica, Cristian Bucur, "A comparison between several NoSQL databases with comments and notes", RoEduNet International Conference 10th Edition, IEEE 2011: Networking in Education and Research, Chengdu, pp. 474-479.

[13]Y Li, S Manoharan "A performance comparison of SQL and NoSQL database" IEEE, 2013

[14]Jing Han, E Haihong, Guan Le, and Jian Du. Survey on NoSQL database in Pervasive computing and applications (ICPCA), 2011 6th international conference on, pages 363–366. IEEE, 2011.