Visualizing Class Diagram Using OrientDB NoSQL Data-Store

Thesis submitted in partial fulfilment of the requirements for the award of degree of

Master of Engineering

in

Software Engineering

Submitted By

Sawinder Kaur

(Roll No. 801431025)

Under the supervision of:

Dr. Karamjit Kaur

Assistant Professor

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

THAPAR UNIVERSITY

PATIALA – 147004

June 2016

CERTIFICATE

I hereby certify that the work which is being presented in the thesis entitled, "Visualizing Class Diagram using OrientDB NoSQL Data-Store", in partial fulfilment of the requirements for the award of degree of Master of Engineering in Computer Science and Engineering submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of Dr. Karamjit Kaur and refers other researcher's work which are duly listed in the bibliography section.

The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

Sawinder Kaur

(Sawinder Kaur)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

Kchoeng2

(Dr. Karamjit Kaur)

Asst.Prof, CSED

Countersigned by

(Dr. Maninder Singh)

Head, CSED

Thapar University

Patiala

(Dr. S.S.Bhatia)

Dean (Academic Affairs)

Thapar University

Patiala

ACKNOWLEDGEMENTS

I am highly grateful for the assistance of all the individuals who spread their valuable time to help me complete my thesis work in time. Foremost I would like to thank my supervisor Ms. Karamjit Kaur for her constant guidance, in-depth knowledge, immense patience, invaluable time, encouraging words and worthy suggestions that helped me in completing my work. Under her guidance, I explored new areas of research. Her practical work approach helped me expand my horizon of research perspective. She has always been a moral support and true guide throughout my research period.

I am also thankful to Dr. Maninder Singh, Head of CSED, and the learned faculty of my department Computer Science and Engineering for their support. The intermediate deadlines set by them helped me work more efficiently.

I am indebted to my prestigious institute Thapar University, for providing me a learning and progressive environment.

My special thanks to all my friends for their endless support, appreciation and moral boost at times of need. I owe a lot to my family for their emotional support and faith in me, for providing me with an opportunity to be a part of this research programme.

Last but not the least I thank Almighty for providing me enough inner strength to pursue my goals and accomplish my work.

(Sawinder Kaur)

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.

In this proposal, an approach is implemented to extract data in the form of class diagram using Java and then storing and running it in OrientDB to represent in the form of graph (using nodes and edges). The methodology employs universal approach for schema extraction by exploiting the class diagram data exported into XML (GraphML) which is supported by most of the NoSQL data stores, hence making the approach generic. Further user queries are handled and formulated on graph database to make better utilization of the extracted data.

We have used 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.

List of Figures

List of Tables

CHAPTER 1

Introduction

Data is said to be raw facts and statics which are collected together for reference or analysis. Data exists in different forms, seen as text on paper pieces or numbers, bytes and bits which are stored in electronic memory, or can be seen as stored facts in human mind. It is processed as an entity which is atomic in nature and is processed to produce some suitable or useful information. There exists a difference between data and program. A program is called as a set of instructions that explains a task for the computer to perform whereas everything else is data but not a program code. Datalog which is a logic programming paradigm is used as an application for real life problems [23].

Database is seen as a collection of information which is organized to make the data easy to access, manage and update. Classification of databases is done according to types of data content as numeric, bibliographic, images and full-text. The most frequent approach is the relational database which is a tabular database where data is defined in such a way that it can be accessed and reorganized in a number of distinctive ways. A database is called when it can be replicated or dispersed among various points in a network. An object-oriented programming database is one which is compatible with the data being defined in subclasses and object classes [24]. Various softwares or applications have been created today that take raw data either from end users or from other similar databases or applications. DBMS is defined as a computer software program which is designed to manage all the currently installed databases on a system network or hard drive. It helps in administering, creating, altering and querying these databases. Various well-known DBMSs are MySQL [17], Oracle, PostgreSQL [9], CouchDB, Redis [6].

1.1 NoSQL Datastore

Carlo Strozzi gave the term NoSQL was in 1998 to name his lightweight as Strozzi NoSQL that is an open-source relational database which has not exposed the standard SQL interface, but was still seen as relational [3]. As current NoSQL departs from relational model, Strozzi suggested that it should be called more appropriately as

"NoRelational", referring to 'NoREL'. The term NoSQL was reintroduced by Johan Oskarsson in early 2009. An event was organized by him to discuss "open source distributed, non relational databases". As the name emphasize to indicate the emanation of increasing number of non-relational, distributed data stores including open source clones of Google's Big Table and Amazon's Dynamo [33].

Earlier NoSQL databases did not attempt to provide consistency, durability guarantees, isolation and atomicity, contrary to which at present such NoSQL systems as OrientDB exists. NoSQL which is also known as "non relational" or "non SQL" database which provides a mechanism for retrieval and storage of data which is obtained in means other than the tabular form which are visualized in relational databases. Such databases exists since the late 1960s, but has not obtained the "NoSQL" appellation as far as a gush of popularity in the early twenty-first century was set of by the needs of Web 2.0 companies such as Google, Facebook and Amazon [25].

The use of NoSQL database is increasing in real-time web applications and big data. NoSQL systems are also known as "Not only SQL" which highlight SQL- query languages [16]. It provides simplicity of design and has simple horizontal query scaling for collection of machines that was considered as a obstacle for relational databases. The data structures classified as key-value, column, graph, or document are used by NoSQL databases and are different from those which are used by default in relational databases, due to which some operations are executed faster in NoSQL. Many NoSQL stores compromise consistency in favour of availability, partition tolerance, and speed, this will be further discussed in CAP theorem.

Instead of using SQL, NoSQL datastores uses low-level query languages. SQL lacks the ability to execute ad-hoc JOINs across different tables, lack the ability of standardized interfaces and requires big amount of investments in relational databases which already exists [11]. Many of NoSQL databases follow a concept of "eventual consistency" in which database changes are successively extended to all nodes "eventually" within milliseconds so when queries are returned the might not get updated immediately or gave give the result of non-accurate data, this problem is known as stale reads. There exists some NoSQL databases which exhibit lost writes due to which loss of data is seen, hence a remedy also exists i.e such NoSQL systems are seen which provide the concept of write-ahead logging

that helps to avoid data loss. Data consistency is seen as a big challenge for both Relational and NoSQL databases [4].

1.1.1 KEY-VALUE DATA MODEL

Key-value is the simple NoSQL data store which is used from an API perspective. User can acquire the value for a key, sets the value for a key or can drop a key from the database. The value is just stored without knowing what is actually stored inside. Key-value data stores always requires primary-key and use a hash table where a pointer points to a peculiar item of data [15]. Values in key based storage of data can be hashes, objects so time taken to run the query for execution becomes less which results in a schema-less and flexible model for modern data which is given the form of an unstructured data, hence gives a better performance and easy scalability will be seen. In the following fig 1.1 it is depicted that how the values are stored in different keys which will use the pointers to point to their original values.

KEY	**VALUE**
K1	XXX,ZZ
K2	SSS,5674,8
K3	BBB,10/04/2016
K4	AAA,CCC
K5	AAAA,BBBB

Fig. 1.1 Key-Value Database

Key-value databases are Riak(Basho), Redis(VMware) [8], Amazon DynamoDB [33], and Couchbase [15]. Table 1.1 explains the difference between Relational and Key-value databases. In relational databases there is requirement to know the data structure before hand which consists of tables and fields inside them so more execution time and memory is required.

Data Structure	A pre-defination of the data structure is seen in RDB with tables which consists of well defined data types.
	A distinct field is seen for every record in where data is treated as a single opaque collection on Key-value datastores.
Memory	Placeholders are used in RDBMS to represent the optional values.

	Very less memory is used in key-value as compared to RDBMS as optimal values are not represented in it.

Table 1.1 Comparison of key-value and relational database

For the updation and querying the value from a database, key-value method is not considered as an ideal. This method is used when designing is required for managing associative arrays, storing, retrieving the data structure. Dictionaries (data structures) consists of many different fields containing data inside each field which consists of records or objects. A key is used to fetch and store the data that uniquely distinguishes the record and is used to search the data within database at increased speed.

1.1.2 GRAPH-BASED DATAMODEL

In Graph NoSQL Database, there exists no restricted format of SQL for storing data in tables and columns, a flexible graphical design implementation is used which helps to achieve the perfect scalability concerns [7]. A pre-defined schema is not required in this data model which leads to easier adaption to retrieve schema evolution. It allows to store entities and relationships amid these entities. Relations are defined on the arcs drawn between the nodes known as edges which have their own properties [4]. Edges are directional and nodes are framed by relationships , through which interesting relationships are found among the nodes [4].

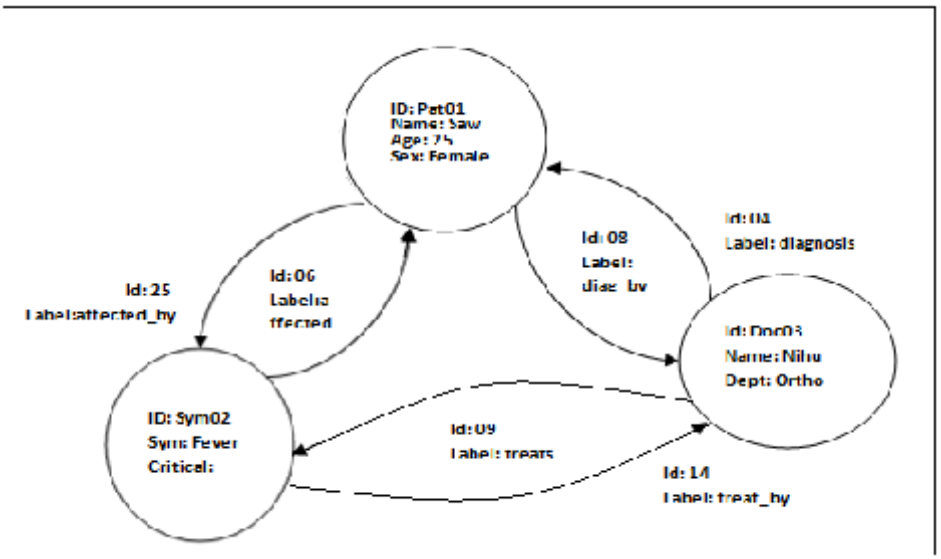


Fig. 1.2 Graph oriented Data Model

The relation between nodes is shown in fig 1.2 which depicts how person with given Id, symptom and situation is related to which Doctor with given department. The graphs can be correlated and interpreted in different ways. Mostly, during storage of a graph-like structure in RDBMS, a single type of relationship is attained, when another new relation is added to the existing structure then lots of data movements and schema changes are viewed which can be easily achieved with the use of Graph databases. According to the traversal user wants, a relational database is to be modelled beforehand. During traversal if data changes then time required to make changes in schema using relational database is more due to use of complex joins in it whereas in graph databases, traversing the joins or relationships is very fast [5]. The relationship between nodes is not computed at query time indeed is 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 InfiniteGraph [5], Neo4j, OrientDB [20], ArangoDB.

1.1.3 DOCUMENTED ORIENTED DATA MODEL

Documents are the prime idea in document data store [19]. The retrieval of the documents from document oriented data model is in any one of the given forms- XML, JSON, BSON etc. The documents are self-defining and consists of map, scalar values, collections forming a hierarchical tree like structure. For example searching for all such documents where "City" name is "Patiala" which would result in getting all such documents which are connected with any "3 Storey Office" that is in developed in that city.

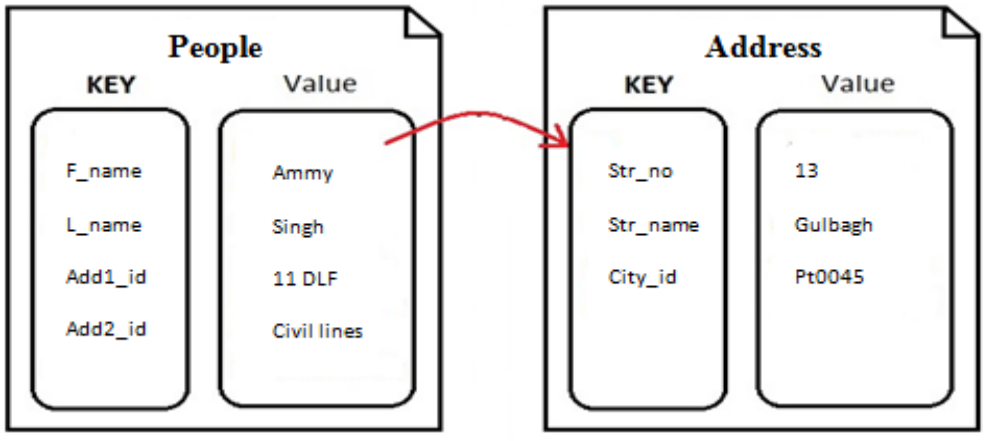


Fig. 1.3 Document oriented Data Model

The above Fig 1.3 describes two documents people and address and an arrow is shown how people document data having values stored in keys is related to address document data values which are also stored in keys. Databases that are styled in documented data model are schema-less so addition of fields become easier.

Data processing	Data is contemplated to be inherently fuliginous to the database in key-value data store.
	System depends on internal structure to extract metadata in document oriented data store.
Data storing	Relational databases 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 1.2 Comparison of Key-value, Document and Relational database

Table 1.2 shows how the storage is done in both Relational and document database is distinguished [28]. Apache CouchDB and MongoDB are popular examples of a document store. JSON is used in CouchDB to store data, JavaScripts its query language with the help of MapReduce [10]. Horizontal scalability, flexibility, high availability was a new challenge for which MongoDB is designed to manage the semi-structured data [7]. MongoDB has real time applications in gaming, archiving, content management systems and mobiles [22].

1.1.4 COLUMN ORIENTED DATA MODEL

Wide-table data stores also called as column oriented data stores, where data is stored in cells collected in columns of data instead of rows of data [13]. Columns are logically grouped into column families. During the definition of schemaor at runtime, a virtually unlimited number of columns [35] are created. In RDBMS write and read are performed using rows whereas in column osriented approach such operations are performed using columns [36]. In comparison, to get faster results for accessing, searching and data aggregation, data must be stored column-wise whereas if want insertion, updation then data must be stored row-wise [18].

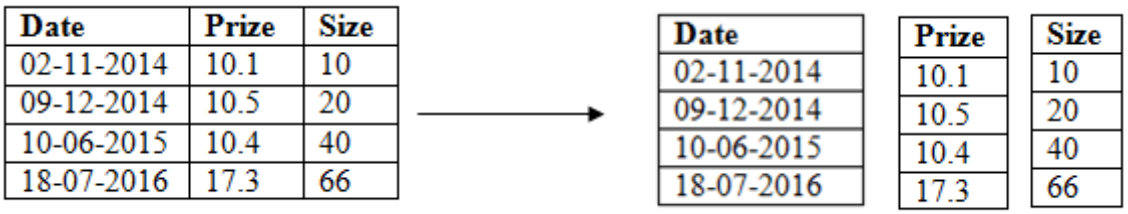


Fig 1.4 Column oriented Data Model

In Fig 1.4, it is shown that how table data is stored in column oriented data form. Table data is collected and stored row wise which takes more time to calculate the sum of price or size whereas in column oriented data model data is stored column wise and calculating the sum of individual prize and size takes less time. A single row is stored as a continuous disk entry in Relational databases [14]. All the data in the cells corresponding to a column is stored as a continuous disk entry making the access and search at faster speed whereas in row database different rows are stored on different locations on disk. For example: To query the titles from a bunch of a million papers published in IEEE will be a hectic task in case of relational databases as items have to taken out by visiting their locations. On other hand, with making just one disk access, title of all the items can be easily attained within few milliseconds according to the data retrieved. Some of the popular open source column-oriented databases are HBase [15], Hypertable [8] and Cassandra [19]. Hypertable and HBase are derivatives of BigTable where as Cassandra takes its features from both BigTable and Dynamo [33].

In this chapter, we have discussed the classification of Big Data [2] which uses NoSQL databases. Among most industries, established competitors want new strategies to capture, compete and innovate their value in market. Data is increasing in every field and is used in critical examinations as in healthcare to analyse the health outcomes of pharmaceuticals. With the help of Big Data new growth in opportunities and various variety of storing data with different velocity and volume will be seen [1]. All four categories of NoSQL – Key value datastore, Graph datastore, Column datastore, Document datastore have been discussed [30]. With use of these databases it is possible to analyze your data and get answer within few milliseconds.

In next chapter, we will be discussing about what actual survey has been done in the field of NoSQL database. What are the comparisons being done between different

databases. Which author has used what type of database and what are the concusions withdrawn by that author. Classification being done with respect to performance, integrity, reliability, interoperatability, query complexity, security. Advantages and disadvantages of using Relational database [27], comparison with Relational database and why to use NoSQL databases.

CHAPTER 2

Literature Survey

NoSQL is an open source a next generation database which addresses these properties: non-relational, non-ACID, distributed, schema-less [12]**.** The actual reason behind the flourishment of NoSQL has been modern or new web-scale databases. Rise of NoSQL databases are challenging the influence of relational databases that has outshined the software industry for longer period [15]. An impedance mismatch is seen between the in-memory data structures and the relational data structures. Without making any conversion from in-memory structures to relational structures, NoSQL database permits the developers to develop the software application [22].

Class diagram with OrientDB helps to store the state of the system [20]. A classification of NoSQL databases is done in this chapter. Advantages and disadvantages of using both relational and non-relational databases have been discussed by the authors. Main emphasis was laid on how to use the OrientDB in the practical working environment. Various modifications that can be brought to facilitate improved performance and superior results were being presented. The design and approach for using NoSQL in the cloud computing environment was being proposed. The basic characteristics, architecture and traits of the NoSQL database are discussed in detail. Features like data models and querying tendencies and abilities of the NoSQL database are zoomed in to a greater extend. Comparative analysis was being performed between a commonly used relational database and a common type of NoSQL graph called Neo4j. Main area of concern of this study is to develop transparency about the emerging non-relational databases by analysis them for various limitations, advantages, features, scope and ambiguities

Projects need some extent of run-time composition where user could compose the rules for the data structures to be 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.

2.1 State of the Art

In this section, we have discussed various ideas and implementations being done by recognised authors. The motivation behind choosing the topic of implementing Class Diagram with OrientDB. Related work to thesis, surveys being already done on NoSQL datastore.

In [23], Stefano Ceri, Georg Gottlob and Letizia Tanea proposed a database query language which was termed as Datalog. Datalog was introduced as a logic programming paradigm that was invented for dealing with the relational database. Through this dissertation main emphasis was laid on how to use the Datalog in the practical working environment. Various modifications that can be brought to facilitate improved performance and superior results were being presented. The future scope and applicability of Datalog at practical level for real-life situations was envisages upon.

In [24], AM Keller, R. Jensen and S. Aggarwal described the technical faults being associated while building object oriented applications (OOA) with the help of relational databases. An application development tool namely Persistence was proposed to overcome the above discussed problem. With the help of this tool C++ applications were merged with the relational database using an automatic code generator.

In [25], Jaroslav Pokorny discussed the role of NoSQL in context to cloud computing. Through this research the design or approach for using NoSQL in the cloud computing environment was being proposed. The basic characteristics, architecture and traits of the NoSQL database are discussed in detail. Features like data models and querying tendencies and abilities of the NoSQL database are zoomed in to a greater extend. Thus, the main focus of this study was to present in depth details about the different modelling methodologies available for designing the NoSQL databases.

In [19], Renzo Angles and Claudio Gutierrez performed an extensive survey on graphical databases which are being described as the data structures used for the schemas that are being modelled as graphs and involved the use of graph oriented operations and type constructors for the manipulation of data. The factors leading to the reestablishment of these graphical databases in the present working environment are being discussed. Through this study graph databases are discussed on the basis of its data structure requirements, querying capabilities, the usage of integrity constraints and the requirement for database modelling techniques.

In [26], Michael Armbust, Nick lanham, Stephen Tu, Armando Fox Michael, J. Franklin, David A. Patterson designed a new database language PIQL, Performance Insightful Query Language in order to deal with complex queries in the distributed key store environment. With the use of the proposed technique the developer is entitled with the flexibility of dealing many queries but with restrictions seen on the count of permittable input output operations which can be executed on the database.

In [27], Chad Vicknair, Michael Macais, Zhendon Zhao, Xiaofei Nan, Yixin Chein and Dawn Wilkins dissertated upon the non-relational database namely NoSQL. Comparative analysis was being performed between a commonly used relational database and a common type of NoSQL graph called Neo4j. The disadvantages of the relational database were brought forward using this comparative study which in turn formed the basis of development of software system incorporating the use of novel technology called Big Table. Big Table was defined as distributed data storage software usually designed for managing very large size (of the range of petabytes) structured data across numerous servers. The various issued that can occur while dealing with Big Table were highlighted. Besides this the advantage of dynamism introduced with the use of Big Table for managing the design and formatting of the database was also discussed with the help of a data model generated by using Big Table.

In [3], Leavitt and Neal performed extensive survey in order to judge the functioning of non- relational databases in accordance to the transitions that were expected from these database systems. In other words the extent up to which the promises made-up on their inception are being successfully fulfilled in the practical working environment are being studied. The advantages of these databases over the age old relational databases are

elaborated. Thus, the main area of concern of this study is to develop transparency about the emerging non-relational databases by analysis them for various limitations, advantages, features, scope and ambiguities regarding these new databases among the user and developer.

In [16], Bogdan Tudorica, Cristian Bucur made an attempt to comment upon various NoSQL systems. In order to reach to a genuine inference a comparison based on multiple criteria was carried out between different prevalent NoSQL databases. In depth analysis was performed in order to determine which among the existing solutions available for NoSQL from its inception onwards genuinely leads to the promised outcomes that yield to better performance.

In [37], Jing Han, E Haihong, Guan Le and Jian Du performed survey on NoSQL database in persuasive computing environment. The incompatibilities of the conventional databases for large scale and highly concurrent applications that lead to the advent of the alternative Databases like NoSQL were being discussed. The basic details related to the background, features, capabilities of NoSQL were being envisages upon. CAP theorem was illustrated as the basis of classifying the NoSQL.

In [28], Shalini Batra, charu tyagi brought to light the various shortcomings in relational database and discussed upon how these drawbacks paved the way for development of graph databases. An elaborative comparison is carried between famous graph database,Neo4j and the most commonly used RDBMS, MySQL. The comparative results were validated by using different benchmark functions. Through the simulation results it was inferred that the graph databases are better in terms of query retrieval time, speed, flexibility and even scalability.

In [29], Mike Buerli highlighted the various applications and implementations obtainable by using the graph databases. The various pros and cons that lead to such immense popularity of these dynamic databases are being discussed. The further areas of research, the open issues that are yet to be resolved in the field of graph database have also been enumerated. Vivid kinds of algorithms, query languages, paradigms that can be considered as the base for implementing graph database at the application level are being considered.

In [30], Karamjit Kaur and Rinkle Rani carried out a detailed analysis on the Internetage databases which are also called as NoSQL databases. The urgent requirement for shifting from relational to non-relational databases which are schema less was being discussed. Detailed survey upon the concept, syntax, types of NoSQL (document, column oriented, key-value pair, graph database) was carried forward.

In [31], ABM Moniruzzaman, Syed Akhter Hossain surveyed upon the concept of Big Data and the various factors responsible for its rapid evolution. The various alternatives to the conventional databases namely NoSQL, New SQL and search based were enlisted. Elaborative study was carried out for NoSQL by discussing in detail its classification, characteristics and evaluation of its performance in the environment employing the use of Big Data.

In [22], Y Li, S. Manoharan validated the fact that the presently used Big Data, NoSQL databases are better than traditional SQL databases. The comparison between the two databases is carried forward in the domain of key-value stores. An abstract key-value store was developed on which basic database operations like read, write, delete and instantiate were tested for both the databases. Through the simulation it was concluded that there are only some NoSQL databases that are better than basic SQL databases.

In [17], Grolinger, K. Higashimo, W.A. Tiwari, A. Capretz focussed upon the storage capabilities or the storage techniques being offered by the cloud environment. The alternatives to basic databases like NoSQL and NewSQL that supports extensive storage are reviewed. The goal of this paper is to study the storage aspect of cloud computing system incorporating the use of NoSQL and New SQl database with the hope of providing a clear vision and guidance to the practioners dealing with these advanced databases. The challenges and opportunities are linked with the usage of non-relational database together with cloud computing platform is briefly discussed.

In [18], O Hajoui, R Dehbi, M Talea discussed in heaps and bounds the concept of Big Data. The various solution to Big Data were classified as NoSQL, NewSQl and search based system. These systems were compared on the basis of performance, integrity, reliability, interoperability, query complexity, cloud support and security. A multi-criteria analysis ROC was being applied to calculate the score of each system. It was concluded that the choice for a king of non-relational system was solely dependent on the

requirements of the user. Relative to the future research scope development of the platform that supports interoperability among the systems was encouraged.

In [2], Chris Snijders, Uwe Matzat, Ulf-Dietrich Reips discussed the stream of Big Data involving the use of both different kind of offline and online networks. Through the analytical experimentation it was stated that power-law was followed for the purpose of distributing the nodes in empirical networks

S.No.	Year	Database	Description
1.	1989 [23]	Datalog with Relational database	Datalog was introduced as a logic programming paradigm that was invented for dealing with the relational databases
2.	1993 [24]	Object Oriented with Relational database	Technical faults were described while building object oriented applications (OOA) with the help of relational databases
3.	2005 [25]	NoSQL with cloud computing	The design or approach for using NoSQL in the cloud computing environment was being proposed
4.	2008 [19]	Graphical database	Graphical databases were described as the data structures used for the schemas that are modelled as graphs
5.	2009 [26]	Performance Insightful Query Language	A new database language PIQL (Performance Insightful Query Language) was designed in order to deal with complex queries in the distributed key store environment
6.	2010 [27]	Graph database (Neo4J)	Comparative analysis was being performed between relational database and a NoSQL graph called Neo4j. The disadvantages of the relational database were brought forward.
7.	2010 [3]	Non Relational database	Survey to judge the functioning of non-relational databases in accordance to the transactions
8.	2011 [16]	NoSQL database	Analysis was performed to determine which solutions are available for NoSQL that leads to a better performance.
9.	2011 [37]	Classification of	CAP theorem was illustrated as the basis of

		NoSQL database	classifying the NoSQL
10.	2012 [28]	Neo4J and MySQL database	Comparison is carried between famous graph database,Neo4j and the most commonly used RDBMS, MySQL
11.	2012 [29]	Graph database	Open issues that are yet to be resolved in the field of graph database have also been enumerated
S.No.	**Year**	**Database**	**Description**
12.	2012 [30]	Big Data	The stream of Big Data that involves the use of both different kind of offline and online networks were discussed.
13.	2013 [31]	Relational and Non-Relational database	Requirement for shifting from relational to non-relational databases which are schema less is being discussed
14.	2013 [22]	NoSQL and its categories	Elaborative study for NoSQL is discussed in detail, classification, characteristics and evaluation of its performance in the environment employing the use of Big Data
15.	2013 [17]	Key-value Data store	An abstract key-value store was developed on which basic database operations like read, write, delete and instantiate were tested
16.	2013 [18]	NoSQL and New SQL database	Study the storage aspect of cloud computing system incorporating the use of NoSQL and New SQL database with the hope of providing a clear vision and guidance
17.	2015 [2]	NoSQL and NewSQL database	Systems like NoSQL, NewSQL were compared on the basis of performance, integrity, reliability, complexity, cloud support and security.

Table 2.1 A review of literature

In Table 2.1 different research papers have been discussed. Most of the emphasis is put on research papers that are related to Survey of NoSQL. In this chapter we have discussed the work which has been done till date on NoSQL databases. Broadly usage of Graph database has been discussed which is part of a real-time system, the data invariably need to be in

memory anyway. The algorithms involved in such a way that hundreds of bits of data is executed in milliseconds. Advantages of using Graph database which is significantly simpler and more expressive that those of relational databases, built for use with transactional (OLTP) systems.

In next chapter problem statement for thesis work has been discussed. Extraction of class diagram data in Java API using OrientDB and then representation of class diagram in graphical format has been discussed.

CHAPTER 3

Problem Statement

A class diagram which is a static diagram is converted in a graphical format. Class diagram basically represents the static design of an application. Class diagram is not only used for describing, visualizing, documenting and constructing various aspects of the software application in a system. The constraints and operations imposed on the system describes the attributes and operations in a class diagram. The class diagrams are the only UML diagrams which can be mapped directly with object oriented languages as are widely used in the modelling of object oriented. Class diagram is seen as group of collaborations, constraints, classes, interfaces and associations. It is static in nature and is also known as structural diagram. With changes in Class diagram at later stage due to change in requirements the whole system requires the reconstruction and hence delay to deliver the software application is seen, also more efforts per person are required. To overcome this limitation a class diagram was required to be converted in graphical format where any alter in the format will not reflect any distortion in previous built design.

NoSQL is a product of handling issues like scalability, complexity and performance and provide enhancement over traditional relational databases. Objective is to create a class diagram in the form of graph using the Java API OrientDB. To achieve the objective eclipse Java8 will be used. A maven project with two packages will be created to retrieve the class diagram data. The relation between the classes will be depicted and how the polymorphism is used between these classes. Aim is to extend the class diagram by adding information about method or by purporting the meta-data in such a way that

addition and deletion of nodes may not affect the other classes is class diagram. Using Relational databases complexity is seen very common cause for making the queries difficult to understand for users so use of OrientDB to remove the problem of complexity. Response time will be compared to measure the performance.

CHAPTER 4

OrientDB

OrientDB is a second generation Distributed Graph Database. OrientDB is the first Multi-Model Open Source NoSQL DBMS that holds the integration of flexibility of documents and power of graphs into single scalable high-performance [13] operational database. Multi-master and sharding architecture are the features which first generation database lacks which Big Data demands. Relational databases are flexible database for the use case diagrams. It has better performance as parts of trees and graphs can be traversed in just a few milliseconds, uses superfast pointers between records, is fast on both write and read operations. When writes are performed, storage upto 120,000 records per second can be executed. OrientDB supports relationships rather supporting costly joins. Even for a Document based database, the relationships are handled as in Graph Databases with direct connections within records. The entire tree or graph can be traversed within few milliseconds. OrientDB supports all three types of schemas-schema-less (not predefined), schema-full (fixed data structure) and schema-mixed (either can be fixed or variable structure) modes [10] and provides the support SQL amongst the query languages.

Class diagram with OrientDB helps to store the state of the system. Almost for every system a logical information model or domain model is framed. These models are used to describe what type of information the system should maintain. The models are shaped in a state to build an object-oriented class diagram, typically in UML. Classes with their properties and association are represented in a model. There exists many databases that show an impedance mismatch during mapping the canonical model. Relational model provides no support for relationships which have to be mapped into keys, inheritance and

polymorphism. In graph data-stores 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, mapping finishes all impedance mismatch as object becomes a vertex, complex properties are handled by documents, provides external support for relationships, inheritance and polymorphism.

4.1 Features of OrientDB

It is important to understand the advantages, weakness and trade-offs among the popular Relational databases. It has become difficult for developers to compromise in flexibility and speed of Database management system products if they want to satisfy requirements of their clients. Hence**,** OrientDB was born which is known as the first Multi-Model Open Source NoSQL DBMS which integrates both strength among graph and flexibility among documents.

1. Better Speed

It has better performance as parts of trees and graphs can be traversed in just a few milliseconds, uses superfast pointers between records, is fast on both write and read operations. When writes are performed storage upto 120,000 records per second can be executed. OrientDB supports relationships rather supporting costly joins [10].

2. High scalability and reliability

It is only the OrientDB which supports both sharded and multi-master architecture, so all the servers are masters. In Fig 4.1 a linear relation between throughput of operations per second and number of servers is shown, with increase in number of servers, throughput operations per second increases.

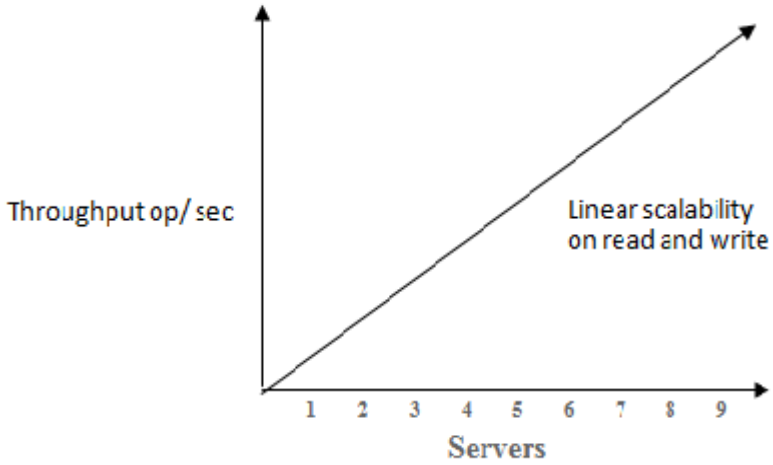


Fig 4.1 Elastic Linear Scalability

The master seen in master-slave architecture can act as a bottleneck but with OrientDB, which is a distributed database does not relies on only single server and solves the problem of bottleneck. The global sum of the throughput is calculated by computing the sum of all the throughput of all the servers. If any extension to the network is required then simply addition of server node in the network will be done and then it automatically joins the existing distributed server with zero configuration. The transactional engine of OrientDB can run in distributed systems supporting many billion records .

3. Distributed Reliability

It is observed that most of the NoSQL products have better performance and scalability but lacks behind in achieving the property of reliability. As NoSQL focuses to speed up the cache. Hence, OrientDB came into existence which holds all the three properties performance, reliability and scalability and also restores the database content after a crash. If any pending transactions are seen then automatically OrientDB will help to roll them back. OrientDB redistributes the loaded nodes or the nodes which caused the failure to the other available server.

4. Provides Cloud Database

There are number of servers which can share the workload across modern distributed data centres [8]. The use of networks on remote servers which are used for storage, managing and processing data is called cloud computing.

5. Easy to Install and Open source

There is no requirement to install OrientDB on any platform, as is written purely in Java. SQL is the widely used standard query language which is also used in OrientdDB. It is Open Source as is publically accessible so design can be modified or any enhancement can be done by anyone.

6. Maturity and Product Stability

The first Multi-model product in the market was OrientDB. Most of the NoSQL products like Neo4j, Key-value datastore etc provide scalability to those applications which already seen on relational databases whereas the complex second generation NoSQL called OrientDB provides better flexibility and functionality as is more powerful and can replace operational databases [34].

OrientDB is basically a NoSQL database that belongs to the family of Graph databases, data is kept in the form of graphs where data is composed of nodes and edges connected to each other [29]. OrientDB is an extremely versatile as it includes Object oriented [16] engines, document database, graph models. It stores and serves records as JSON [10] documents and performs the indexing algorithm (MVRB-Tree). MVRB-Tree is a mix of red-black tree, because of this algorithm it becomes fast to insert/update/delete in the database. MVRB-Tree used Red-Black tree for implementation which helps to maintain the initial speed whereas it maintains a proportion for insert or update process. With the use MVRB-Tree one can see fast retrieval and storing of nodes. Multinational companies like Sky, Lufthansa, Cisco and Ultra DNS are already using OrientDB in its production [20].

OrientDB is considered as a new-generation database as no other database can provide all such features together. Thousands of records can be traversed in milliseconds to any level of depth, it uses simplified SQL syntax as does not uses costly joins, all such features make OrientDB as a game changer in DB market. Performance Insightful Query Language(PIQL) [26] are used for complex queries and give access to the developers to express different queries found on websites by providing strict bounds on the operations to be performed.

4.2 Comparison of MySQL, MongoDB, OrientDB, Neo4j

NoSQL provides the current trend of data management technologies which are modelled to meet the increasing velocity, volume and variety of data. They can store and retrieve data which is modelled other than the tabular structures which are used in relational data-stores.

Table 4.1 shows the that OrientDB is the only database which can overcome the limitations of all three other databases i.e MongoDB, Neo4j, MySQL. OrientDb is the only database which can be implemented in both graph and document databases. Neo4j cannot be implemented as operational database but rest three can be implemented. OrientDB uses SQL like query so to implement the commands is easier as every user is familiar with it and also it does not uses expensive joins. OrientDB is the only popular database which supports all the three schema-less, schema-mix, schema-full variety of data and Object Oriented concepts are also only used in this database.

DBMS				
Features	**OrientDB**	**MongoDB**	**Neo4j**	**MySQL**
Operational database	✓	✓		✓
Graph Database	✓		✓	
Document Database	✓	✓		
Object oriented concepts	✓			
Schema full, less and mix	✓			
SQL	✓			✓
ACID Transactions	✓		✓	✓
Multi-master replication	✓			
Sharding	✓	✓		
Custom datatypes	✓	✓		✓
Relationship traversing	**O(1)**	**O(LogN)**	**O(1)**	**O(LogN)**
Record level security	✓			
Record level recording	✓		✓	✓

Table 4.1 Comparison of NoSQL datastores

4.3 CAP Theorem

It is a basic requirement which describes the distributed system.

Consistency – It means that data is same across the server, from any node one can read/write the data and will get the same data as a result. If a request is put on two servers for same query, both the servers will get same copy of data.

Availability -It means the ability to ingress the server even if a node in the server goes down.

Partition Tolerance – If ability to reach any server gets failed or server crashes, still the entire system continues to operate.

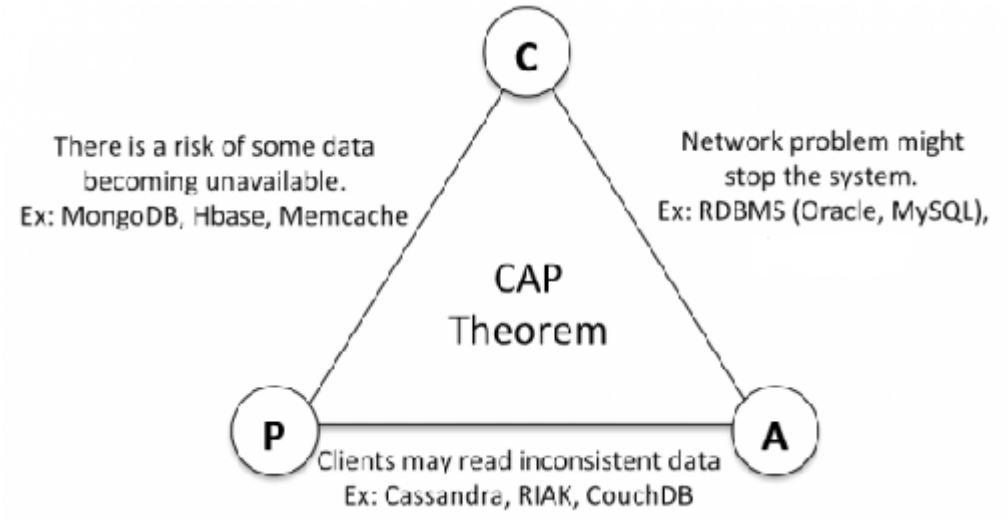


Fig. 4.2 CAP Theorem

Fig 4.2 shows what type of databases support partition tolerance, availability, consistency [12]. Practically it is not possible to have all these three requirements to be met together so a combination of two among three is chosen, it helps to decide what technology is used.

4.4 ACID vs BASE Consistency Model

It is for the relational databases and describes a set of properties that employ to data transactions. ACID [12] uses SQL i.e structured query language, a standard language to which every user is familiar with uses pre-defined tables for storing the information as well uses costly joins, hence the queries become difficult to understand . They store data in spreadsheet like tables having data types and columns strictly defined.

Atomicity – It means all or nothing which means if the transaction takes place then everything during transaction must happen successfully or none of the changes among transaction must take place.

Consistency - The resultant data which has been given as an input, after execution must get the same resultant data. If transactions are performed in account 'A' then so transactions must be performed in account 'B' during money transfer.

Isolation – Visualize that the transactions are running independently, no concurrent execution of transactions will be seen. Hence other users won't be able to view the partial committed transactions.

Durability - Once data is committed, whether any crash or failure may occur in the system will not lead to any malfunctioning within the database.

ACID follows the property of structured database which stores data in the form of table. For reliability and consistency relational databases are designed. The NoSQL database model does not uses the structured relational model in favour of a flexible key/value store approach. So there comes an unstructured approach to data which is called as BASE model.

Basically available - Means availability of the data. If the part of data is not available or any node fails, still the whole data layer stays serviceable.

Soft state – It helps to state that the data which needs refreshing after a period of time. Without any refresh either the data gets perished or gets deleted.

Eventual consistency - Means whatever the updates done to one server will ripple through all the servers, within given time.

As huge amount of data is being designed by prevalent companies, NoSQL databases now seen as matured databases to handle performance and scalability issues. They are now known to put more emphasis on stockpile of much bigger magnitude of data in form of peta and tera bytes. Also they drop the support of joins making it less expensive, they used distributed servers together and stored the data differently.

4.5 OrientDB Concepts

Record is the smallest unit in OrientDB that is used for loading and storing data in database. There exist four types of records as Vertex, Recordbytes, Edges, Document. In Graph data-stores nodes are known as vertex and the line drawn between two nodes represent the edges [31]. When a record is generated by OrientDB, it automatically assigns a unique number known as RecordId.

The Document is the most flexible record type available in OrientDB. Documents are softly typed and are defined by schema classes with defined constraints, but you can also use them in a schema-less mode too. Documents handle fields in a flexible manner. You can easily import and export them in JSON format. When OrientDB generates a record, it auto-assigns a unique unit identifier, called a Record ID, or RID.

4.5.1 Datatypes of OrientDB

In OrientDB, when a class is created, by default classes created are schema-less which means that each record belongs to a class that has different fields of any supported data type [20]. OrientDB does not uses the costly joins which were used in relational databases. OrientDB supports several datatypes which is described in Table 4.2 . It also has many similar datatypes of SQL like Boolean, integer, long integer, short integer, double, long double, float, string.

Sr. No.	Type	Description
1.	Boolean	Handles only the values True and False
2.	Integer	32-bit signed integers
3.	Embedded	Owner contains the record inside it. The records inside the owner has no RecordId.
4.	Embedded List	Owner contains the record inside it. The records inside the owner has no RecordId and the records are reachable using the owner record.
5.	Embedded Set	Owner contains the record inside it. The records inside the owner has no RecordId and the records are reachable using the owner record.
6.	Link	It is one-one relation and links to another record.
7.	Link List	It is one-many relation and links to another record and

		here only the RecordId's are stored.
8.	Link Set	It is one-many relation and links to another record and here only the RecordId's are stored.
9.	Any	Has no determinate type and is used to specify collections of null and mixed type.
10.	Transient	Any value not stored on database.
11.	Binary	Consists of any value as byte array.
12.	Date-Time	Date with exactness up to milliseconds

Table 4.2 Datatypes of OrientDB

4.5.2 OrientDB basic commands

The commands used by OrientDB is similar to SQL commands. The major difference is that in OrientDB no costly operations like joins are applied. OrientDb does not has any pre-defined data structure for the table. In Table 4.3 some basic commands of OrientDB which are most commonly used have been shown with syntax and description. To execute list database different databases are used.

S.No	Name	Syntax	Description
1.	Create	CREATE DATABASE <database_url> [<user> <password> <storage_type> [<db_type>]]	URL consists of mode and path. Mode defines remote mode or local mode. Path defines the actual path to database. User defines the user you wish to connect with database. Storage-type like plocal and memory.
2.	Connect	CONNECT <database_url> <user> <password>	With the help connect having valid user and password one can connect with OrientDB.
3.	Import	IMPORT DATABASE <input file>	When one wish to import the database, to generate an export command a JSON format file is exported.

4.	List	LIST DATABASE	It is used to get the list of all databases.
5.	Export	EXPORT DATABASE <output file>	JSON format is used by OrientDB to export the data. A GZIP algorithm is used within export command to compress the files. When database is exported it is does not gets locked, so read and write operations can be performed on it.

Table 4.3 Syntax of basic OrientDB commands

4.6 Class Diagram

Class diagram is the Unified Modelling Language (UML) and has a static nature which describes the design of a system by displaying the system's attributes, relationships among objects, their methods, their classes and attributes. Class diagram and use case diagram are the building blocks of object oriented modelling concepts. Class diagram is used for detailed analysis modelling for translation of models into a programming code, also is used for general conceptual modelling in a systematic way for the applications [32]. In data modelling, class models are the most frequently used. The classes in a class diagram represent the important elements, the classes to be programmed and the interactions within the application.

Abstraction

It is to provide the important information to the outside world and hide the unnecessary features i.e only the required information in program is seen without any undesirable information.

Encapsulation

Object oriented programming includes the concept of encapsulation which means binding functions and data together that will modify or change the data and will keep both data and functions safe from the interference or misuse from outside world. Example, a capsule

consists of powder which is covered with outside layer so that inside material must not be affected.

Polymorphism

Same functions and operators are given different meanings to be operated separately. The word polymorphism means to have 'many forms'. When hierarchy of classes is seen and relation between hierarchies must have inheritance then polymorphism is seen.

Class and Objects

Object is an entity that has state and behaviour. Example chair, marker, car, pen, table. State represents the value of an object. Class – It is a group of objects that has common properties. By convention first letter of class is in upper case and rest are in lower case. Behaviour represents the functionality of an object. Pen is an object , with name as matrix and colour white which represents its state. It is used for writing (it tells the functionality).

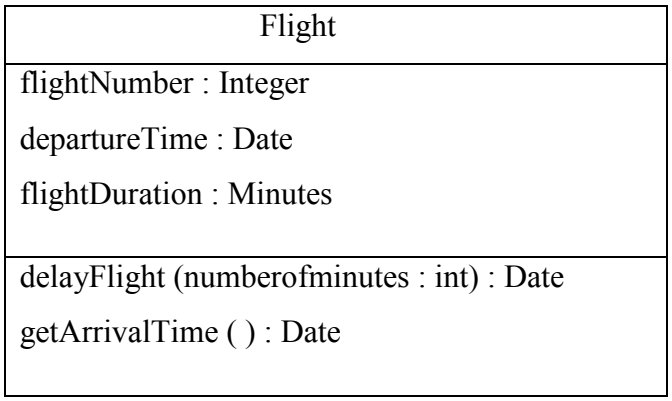


Fig 4.3 Class diagram for the class Flight

In Fig 4.3 a class name Flight is taken which has three attributes – flightNumber, departureTime, flightDuration with two operations as delayFlight and getArrivalTime. Example company is a class then TCS, Vipro etc are their objects. A class diagram is a type of diagram that is static in nature and which represents the design of the system by just visualising the attributes, relations among various objects, system classes and operations. Representation of class diagram is seen in the form of a rectangle which has three partitions horizontally. The top partition represents class name. The middle partition represents class attributes, bottom represents list of class's operations. In class diagram middle and bottom partitions are optional..

Inheritance

It is an ability of one class to inherit the similar functionality of another class or super class and then can add new functionality of its own. In a class diagram a solid line is drawn from the child class with a closed, unfiled arrowhead pointing to a super class. Classes are inherited with the help of Object-oriented programming using state and behaviour from other classes.

```
class PlateauBike extends Bike {
   // new methods and fields are defined here
   // Plateau bike would go here
}
```

In previous example, a bike has become the superclass of PlateauBike, RoadBike, and HillclimbBike. Each superclass can have unlimited number of subclasses whereas each class is allowed to have only one direct superclass, this is seen in Java programming.

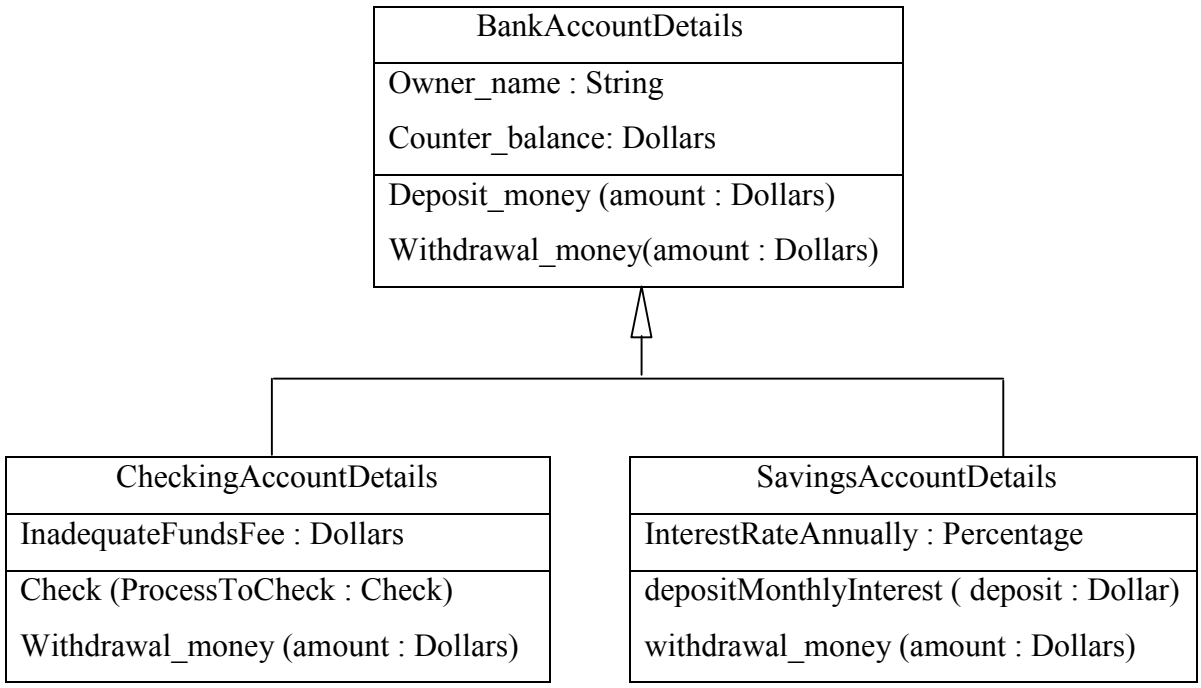


Fig. 4.4 Example of Inheritance using tree notation

In Fig 4.4 inheritance property is followed where a bank account has two classes CheckingAccountDetails and SavingAccountDetails. These two are the inherited classes

which have the properties of their own and can inherit some properties of parent class name BankAccountDetails.

In this chapter, emphasis was put on the features of OrientDB. Basic commands and datatypes used in OrientDB. Comparison was done between OrientDB, Neo4j, MySQL, MongoDB. Complexity for these databases for traversing a relationship was depicted. OrientdDB and Neo4j takes constant time for traversing a relationship so gives the fastest result for a given query. All concepts related to object oriented programming are explained so that. Emphasis was put on UML diagram i.e class diagram.

In next chapter, how together OrientDB integration with Java API works together to achieve the graphical visualization of class diagram is discussed.

CHAPTER 5

Implementation

OrientDB supports JDBC in a similar way as RDBMS supports. OrientDB is one of the NoSQL database which supports a subset of SQL like query language. Java Database Connectivity (JDBC) is an application program interface (API) stipulation for connecting various programs which are written in Java to the data of the popular or required databases. Interface of an application program helps to access the requested statements in Structured Query Language (SQL) which are then wended to the program through which the database is managed. Through this interface results are returned.

5.1 OrientDB-JDBC connectivity

The JDBC driver is registered under the Java SQL Driver Manager. With the use of JDBC driver all the OrientDB database types- memory, plocal and remote etc can be used for integration work. A running database with a table is an important requirement so one can query and modify, install that database which is most suitable. Create the connection, then start the database server by executing the commands if the server is not running, execute the statement, close the connection. Consider an employee table with following fields and

types. In Table 5.1, Employee with id having integer datatype, name field having string datatype, salary field with integer datatype and join date as date datatype.

Sr. No.	FIELDS	TYPES
1.	Id	Integer
2.	Name	String
3.	Salary	Integer
4.	Join_Date	Date

Table 5.1 Employee Table

Create a Schema (table) by executing the following commands.

```
CREATE DATABASE PLOCAL:/opt/orientdb/databases/testdb
CREATE CLASS Employee
CREATE PROPERTY Customer.id integer
CREATE PROPERTY Customer.name String
CREATE PROPERTY Customer.salary integer
CREATE PROPERTY Customer.join_date date
```

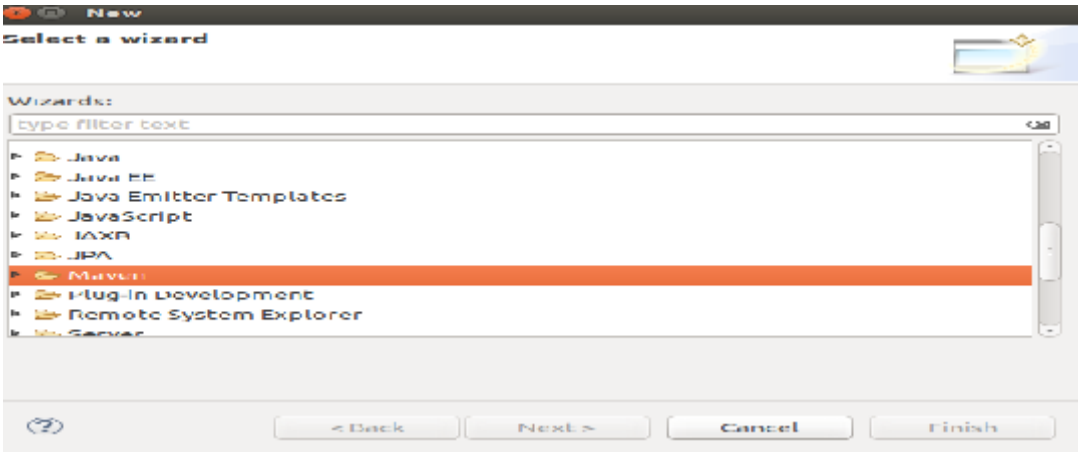
Employee table is created after the execution of these commands. In next section, data of class diagram will be imported in Java through Java API and with support of OrientDB library.

5.2 Import data in OrientDB using Java API

A java application is created using Google Reflections and ASM from where the information is extracted about library OrientDB 2.0.6, then JAVA API is used to create the schema and populate it. Java API is a java application programming interface (API) that is

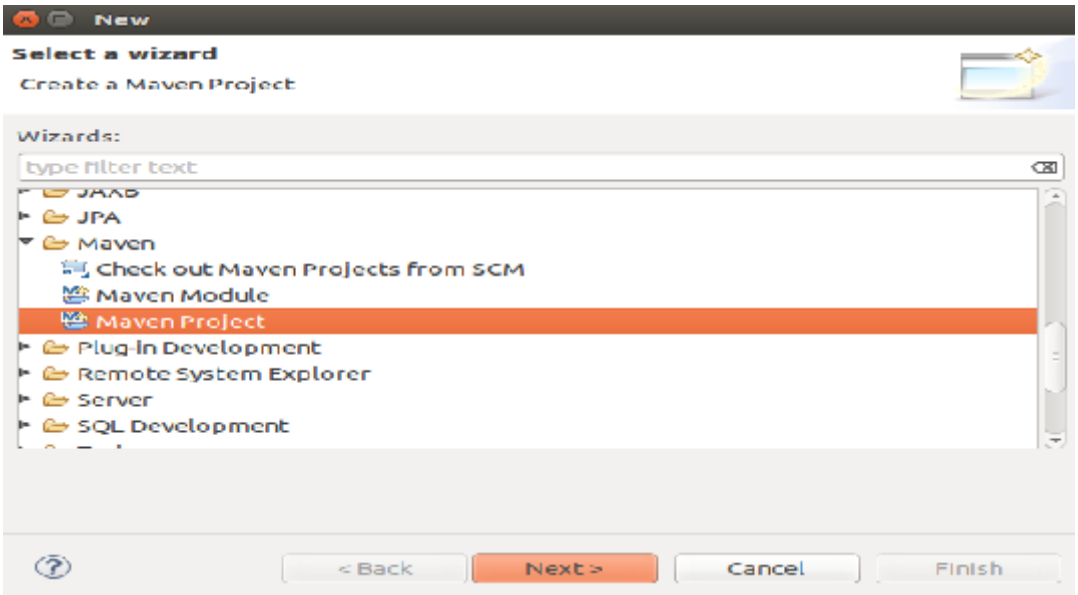
a list of all classes that are part of the Java development kit (JDK). It includes all Java packages, classes, and interfaces, along with their methods, fields, and constructors. These prewritten classes provide a huge amount of functionality to a programmer.

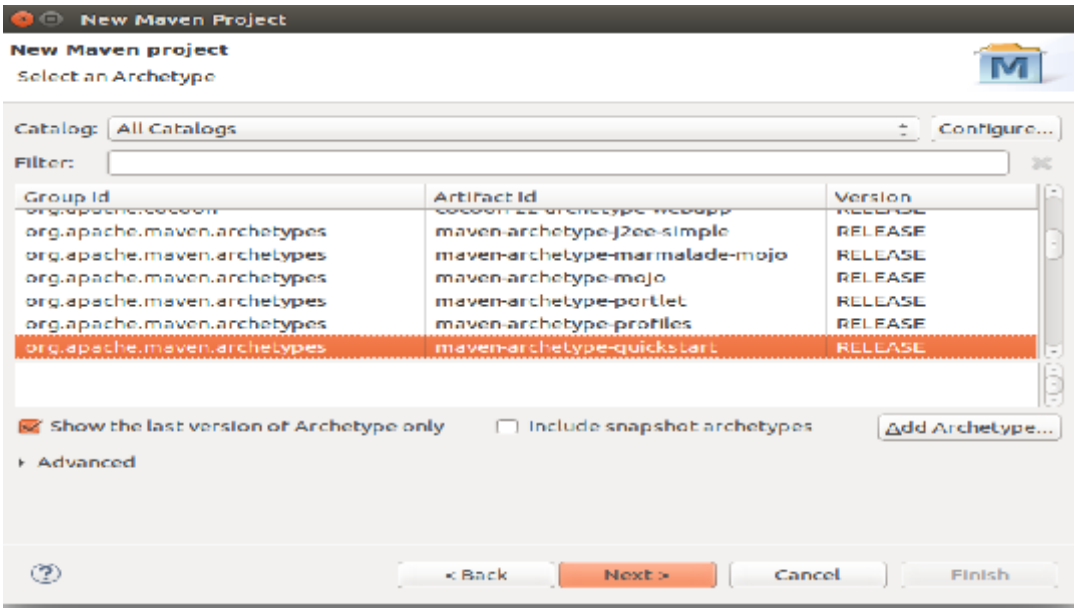
Step1: Select Maven from wizard



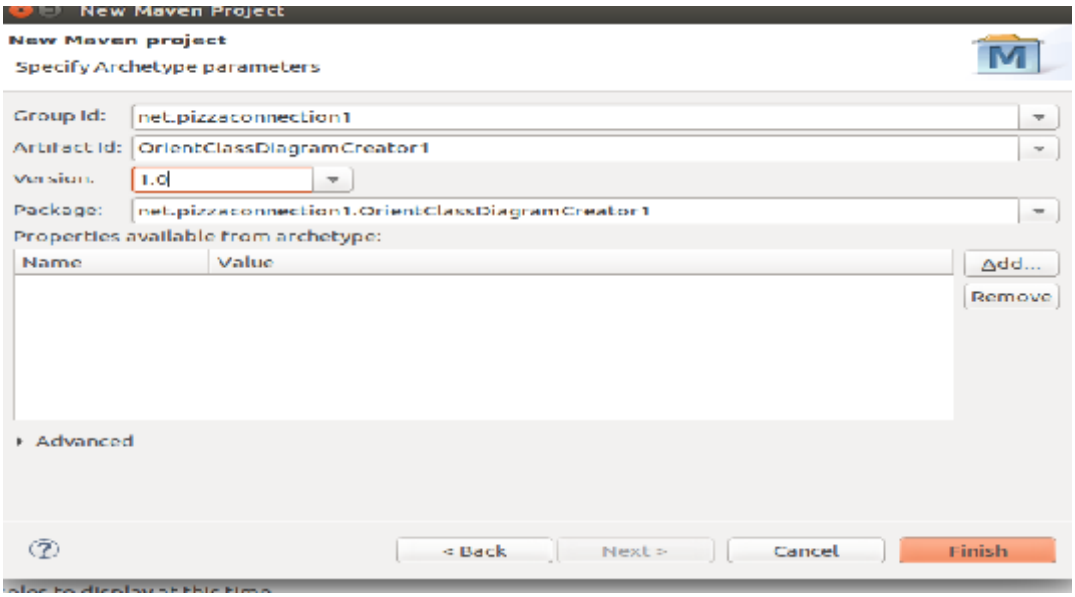
A new Maven project is created in Eclipse IDE. The Eclipse IDE provides support for the Maven. It helps to manage dependencies in the project, helps to update the class path of the project dependencies and it makes ensures that your experience in Eclipse of working on maven is smooth. Eclipse IDE provides an editor of the POM.xml, helps to provide various kind of wizards to import, supports to create new maven based projects.

Step 2: Select Maven Project and all Catalogs in category of Catalog.





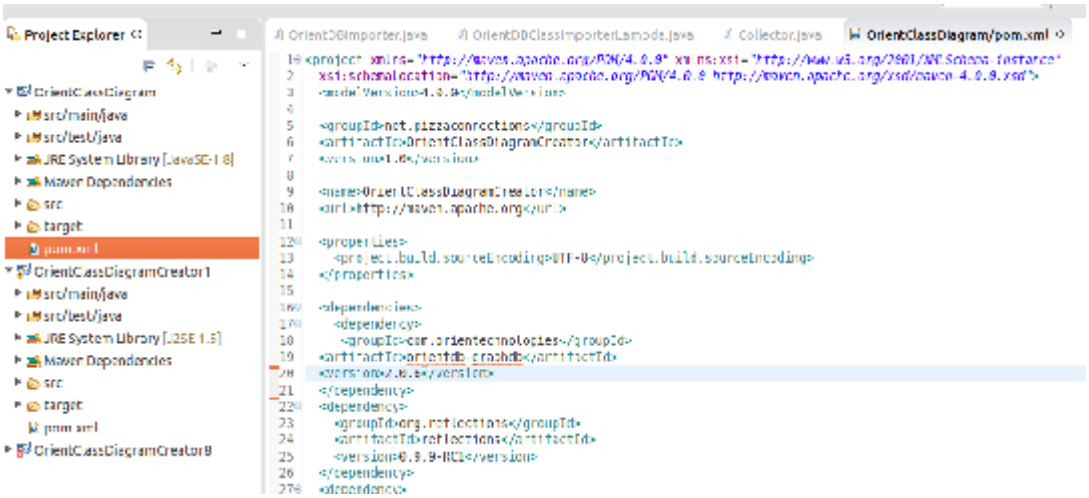
Step 3: Select Group Id, Artifact Id, Version and Package name.



5.2.1 Add Dependencies to POM.xml

POM(Project Object Model) is a XML file which contains the project configuration details used by Maven. POM defines the model of the project ,adds jar files and libraries as required. In maven based development, these jars and libraries are added to the project using pom.xml. In POM all jars and libraries are called as dependencies

Step 1: On creating Maven project an auto generation of POM.xml dependency is seen

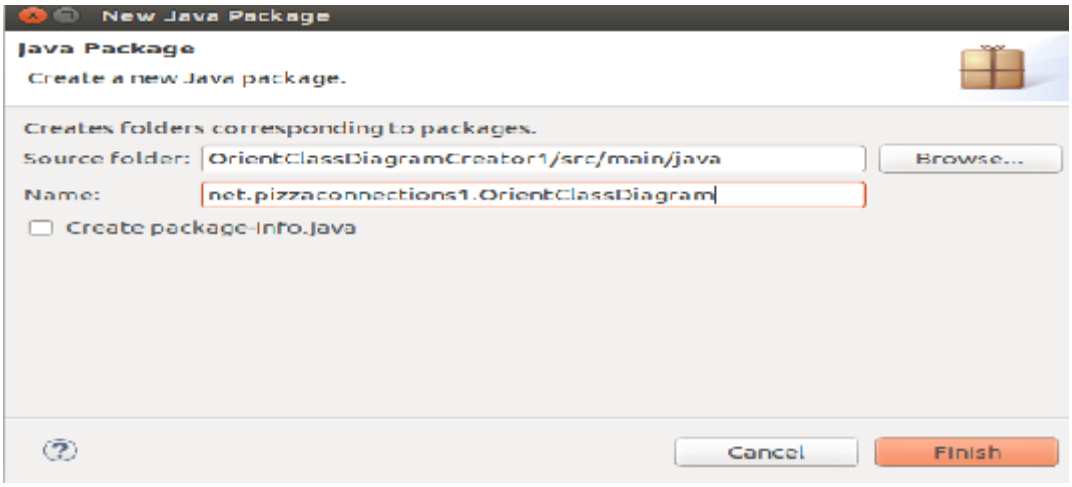


Mostly every project depends upon other projects to run and build correctly without any error, Maven helps to manage the list of projects on which you can depend so that projects can be run without creating any obstacle. Maven also helps to download and provide the links for the dependencies during compilation. The triangle of GroupId, ArtifactId and Version is used to calculate the coordinates for Maven for a specific project which is based on time to separate it as a dependency of the project. The reason behind doing all of this computation is to choose a version which helps to provide all types of dependency declarations. The GroupId and ArtifactId correspond to the co-ordinates of the dependency whereas version is used as a purpose to compute the effective version of dependencies.

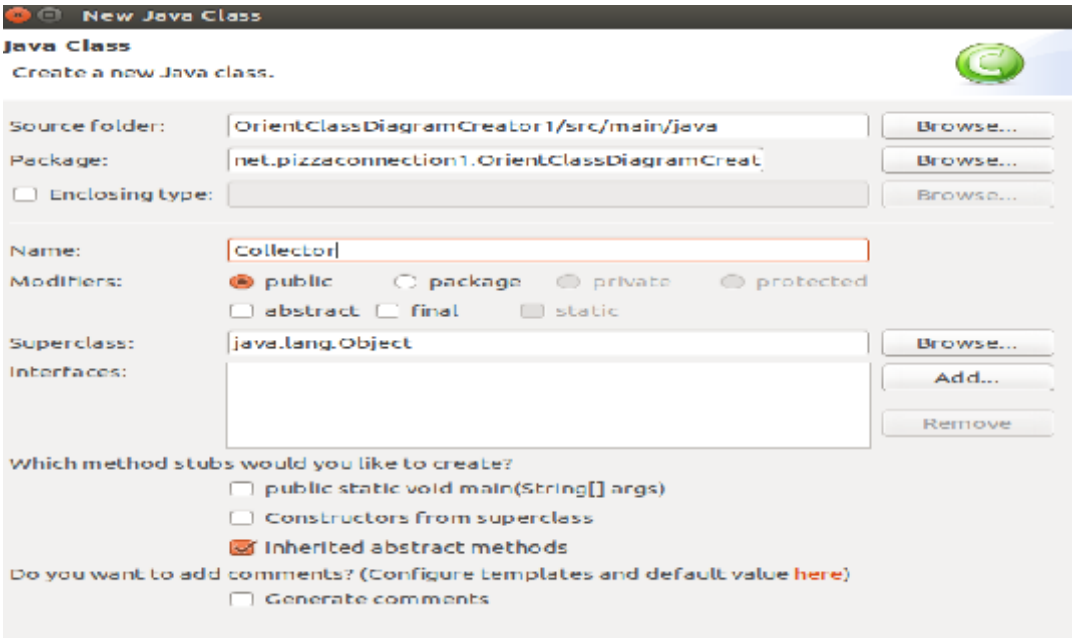
5.2.2 Creation of Collector Class

A collector class is created which helps to find out the classes that are used by a particular class. This is done with the help of ASM library. A collector class is created within the package of the OrientClassDiagram.

Step1 : Choose name of the source folder, package and its superclass



ASM is used for manipulation of java byte code and is an open source. A byte code is manipulated with use of ASM which helps to eliminate the usage of loading the whole class in memory. ASM consists of instructions and each instruction does not has its own separate class instead has constants to represent them , hence it helps to reduce the library size. A new java package is build for the implementation of visualising class diagram with OrientDB. ASM allows a class to analyze and modify a class from bytecode. ACM is used in both Static and dynamic systems and makes it attractive and fast. Classes are dynamically generated using ACM, with the help of ACM an existing class can be modified directly in binary form. It helps to provide the common analysis algorithms so that it becomes easier to assemble the code analysis tool and complex transformations.

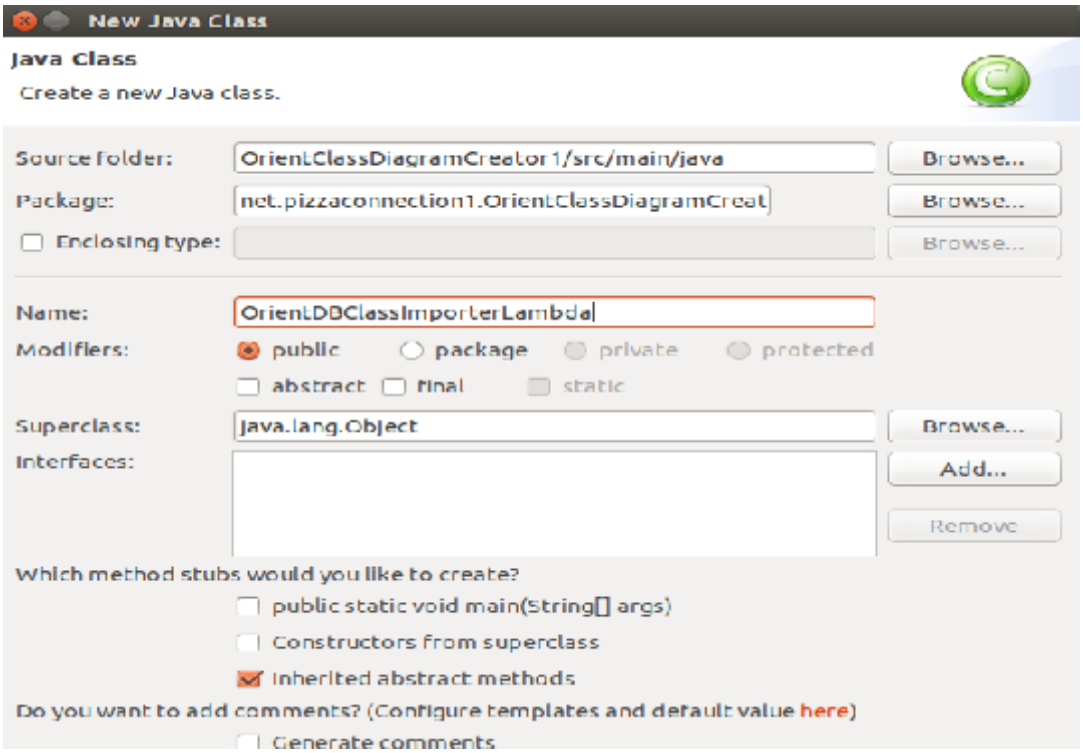


After creation of collector class, code is written in Collector class of Maven project to retrieve the data of class diagram in the form of tree in java which further will be represented to user in OrientDB.

Step 2: Create a new class under maven project in OrientclassDiagramCreator package.

A new class is created named OrientDBClassImporter which will extract all data about OrientDB library 2.0.6 classes, then it will help to import the result into the OrientClass Diagram schema of our database.

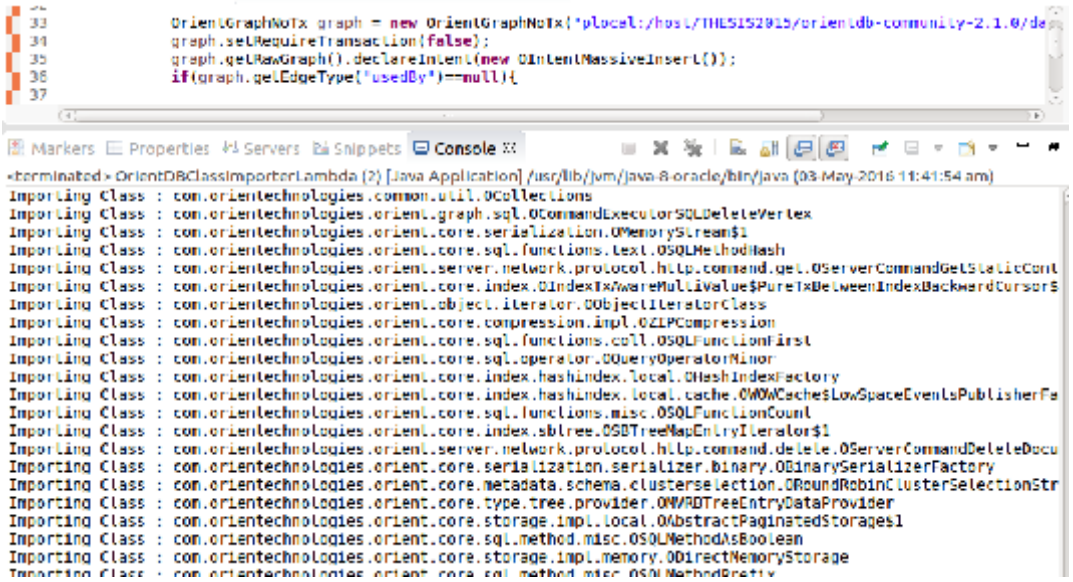
Step 3: Use Import class method



Here the OrientDBClassImporter class uses the importClass method recursively to extract and import information about the class name, the origin package, the origin library and relationships with the possible superclass and used classes. OrientDBClassImporterLambda class code that works same as OrientDBClassImporter class but uses the java 8 lambda expressions.

Till here a maven project is created in Java8 eclipse and POM dependencies are added automatically. Two classes are created and within the classes packages were created and further coding is done to retrieve the class diagram data in OrientDB.

Step 4: Execute the application in Eclipse



Run as Java Application with OrientDBClassImporter or OrientDBClassImporterLambda to import the data.

5.3 Hardware and Software Requirements

This section defines the hardware and software requirements for the evaluated Configuration. For the user to understand what requirements are met during implementation, both hardware and software environment is required.

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

Table 5.2 Hardware Environment

In Table 5.2 all hardware requirements to be met for implementation of the project is shown. RAM which is a main memory, must be of 2GB and hard disk drive required for secondary storage must be 40GB. Internet connection is required as errors done during coding can be sorted using online available codes.

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

Table 5.3 Software Environment

In Table 5.3 Software requirements are shown. Actual implementation of work is done in Ubuntu 14.04 which gives better efficiency for downloading and queries are run easily on Ubuntu. In windows 7 all paper work is done as it provides better convenience to the user. Research paper and thesis documentation was made on word. The programming language used to retrieve the class diagram data with support of OrientDB library is Java8 (eclipse). Connectivity of java with OrientDB was done and after running some OrientDB queries a graphical format of class diagram in OrientDB is represented.

5.4 Flow Chart to Implement Class Diagram using OrientDB

A flowchart is used to make implementation of class diagram in graphical format easier to understand that how the process has taken place. In Fig 5.2 a flowchart is used to explain the steps, firstly a class diagram data is retrieved in Java8 eclipse IDE. A maven project is created which automatically generates POM dependency. A new Maven project is built in Eclipse IDE. The Eclipse IDE provides support for the Maven. It helps to manage project dependencies, helps to update the class path of the project dependencies and make sure that in Eclipse the usage of maven is smooth. A class named importerclass is created to extract the data in OrientDB with the help of OrientDB libraries and Java API. Database of class diagram is connected with OrientDB. In OrientDB studio, different queries are run and resultant response time for different queries is noted down, also we get to know the relationship between superclass and subclass. A class diagram is visualized in the form of graph. Any changes can be made to the classes seen in the form of nodes in graph diagram, the changes in the classes made will not reflect any change in behaviour to other classes. Only the attributes related to the change classes will be affected.

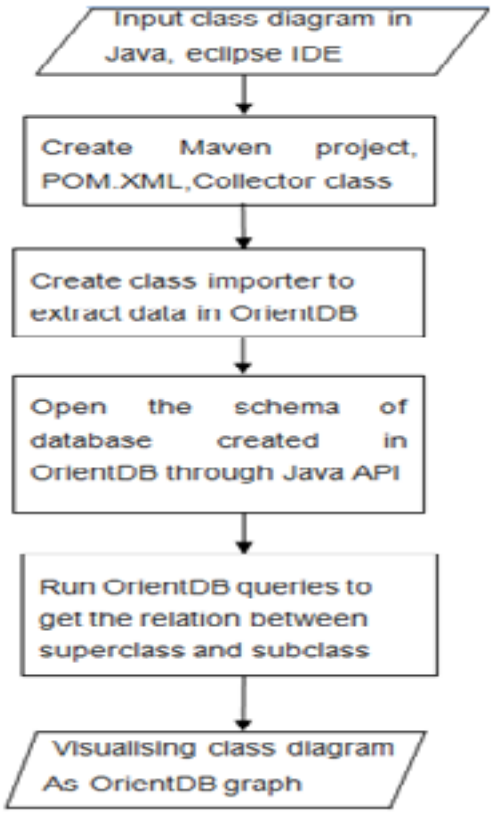


Fig. 5.2 Flow chart to implement class diagram

Now OrientDB studio is opened by making the database connectivity. Run different queries so that a relation between superclasses and subclasses is seen. Finally a graphical format of class diagram is shown with the help of OrientDB. Class diagram with OrientDB helps to store the state of the system. Almost for every system a logical information model or domain model is framed. These models are used to describe what type of information the system should maintain. The models are shaped in a state to build an object-oriented class diagram, typically in UML. Models represent the classes with their associations and properties among classes.

Among many databases some sort of impedance mismatch is seen when mapping is performed in canonical model. Relational models do not support mapping of relationships into keys, polymorphism and inheritance. In graph data-stores, there is also no abutment for polymorphism, inheritance and complex properties introducing new vertices. A document database also does not provide support for polymorphism and provides very limited support for relationships.

.

CHAPTER 6

Running examples and Results

A command line interface is provided by OrientDB. It can provide the help to connect and work with local or remote OrientDB servers. Till now the code in java was written. In this section, we will be discussing how database is connected with Java API. After the database connectivity, OrientDB studio is opened using localhost. Different queries will be run to check the behaviour of class diagram which is to be retrieved in graphical format. It is only the OrientDB which supports Key-value, Object oriented, Graph oriented and Document oriented data models so OrientDB can be used as a replacement for any of the products obtained by any of these categories. The reason behind choosing OrientDB is its support for Multi-model databases which combines all the four models into a single model. In this section, we will discuss how the working of OrientDB takes place, how the queries are run in OrientDB noting their response time and concluding the reason behind the different execution time of queries. With the help of these queries, relationship between classes have will be depicted through graph in OrientDB and their performance have been noted down.

In this section, how the actual implementation of class diagram in OrientDB will take place when connectivity of Java API with OrientDB will be done. The classes called as nodes are connected with each other and what is the relationship between the nodes which is depicted by the joining edges known as arcs between the nodes. Graph will be created that contains a class diagram using the Java API OrientDB. The JAVA APIs are very simple to use, allow to model and populate a graph without need to know SQL commands. Different queries will be run and the time taken to execute the queries in browse editor will be noted. A performance analysis is done in which response time to retrieve the query will be noted down and according to the readings bar graph will be generated to make easier conclusion why the response time of the queries vary.

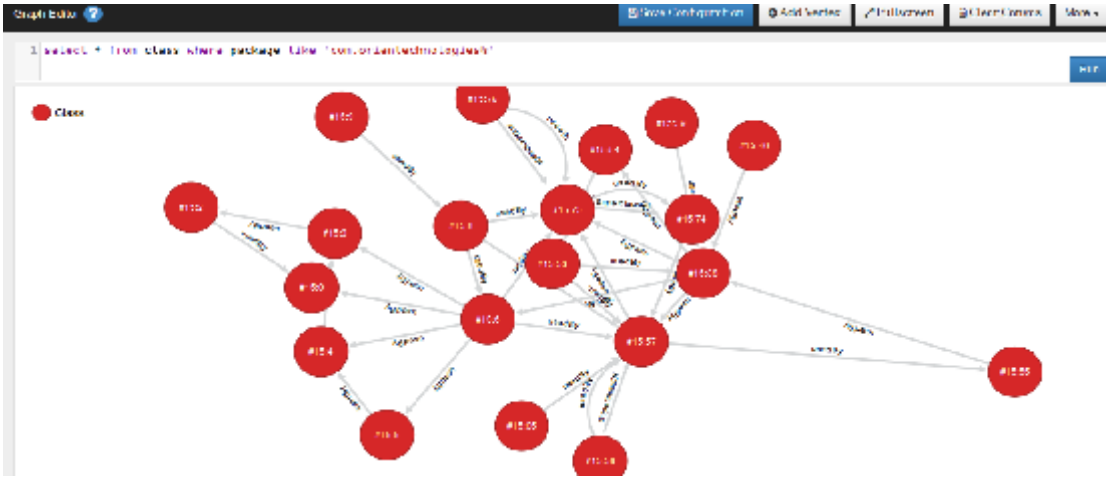
6.1 Analysing class diagram in OrientDB

Start OrientDB server (studio on localhost) and connect to OrientClassDiagram database schema. Write the queries to retrieve the graphical data in graphical editor.

Step 1- Click on graph to execute simple query on class to navigate in graph.

Query 1 Retrieve the class diagram in the form of graph.

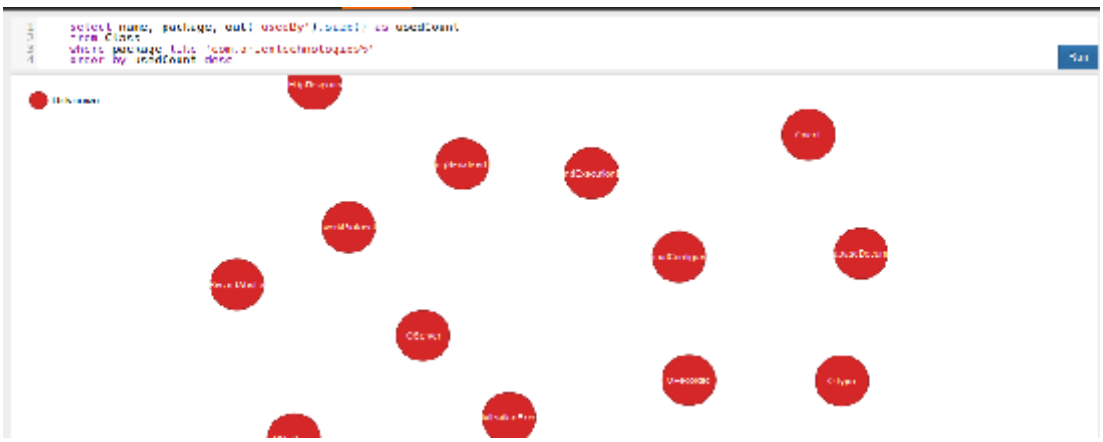
```
select * from class where package like ‘com.orienttechnologies%’
```



In above snapshot one can visualize how the nodes are connected with each other and what is the relationship between the nodes which is depicted by the joining edges between the nodes.

Query 2 Query to understand the OrientDB library class with more usedBy relationship.

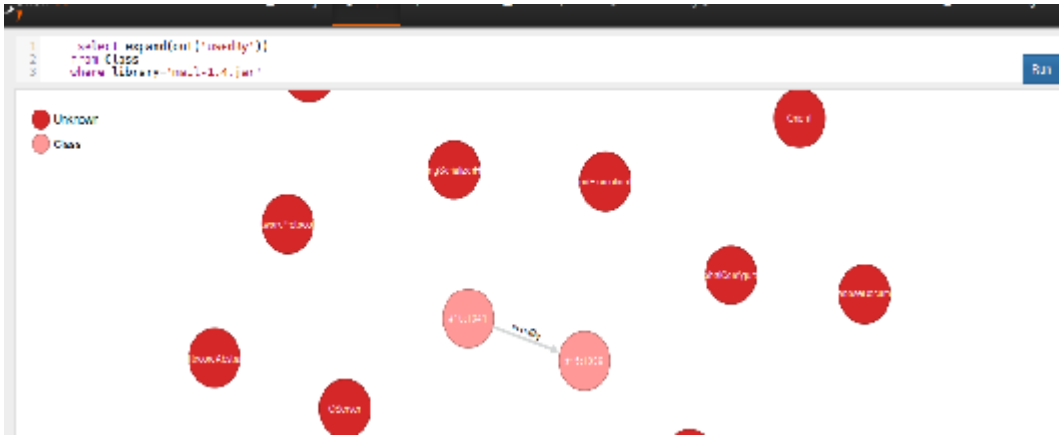
```
select name, package, out(‘usedBy’).size() as usedCount from class where package like ‘com.technologies%’ order by usedCount desc
```



The above snapshot shows the individual identity of nodes.

Query 3 Query that returns all classes that use a particular library.

```
select expand(out('usedBy')) from Class where library='mail-1.4.jar'
```

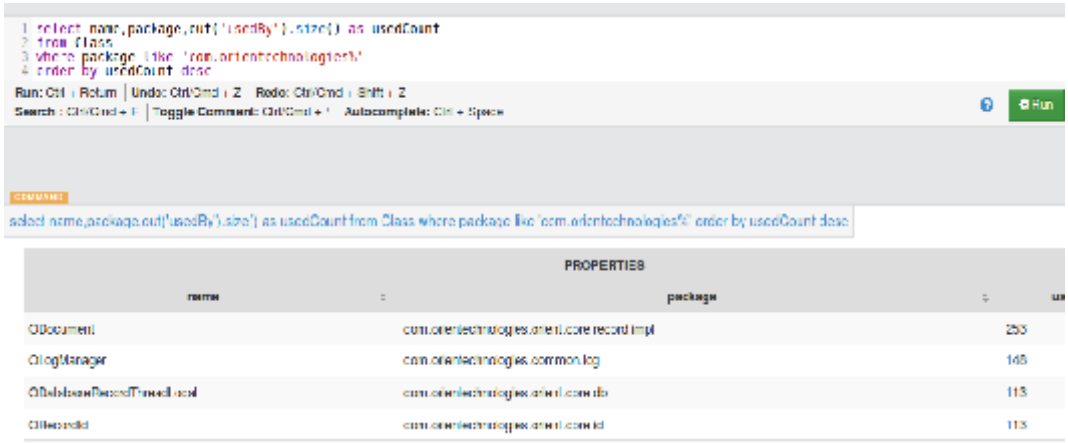


The above snapshot shows the relation between two individual classes i.e nodes which are not connected with the other classes in the graph.

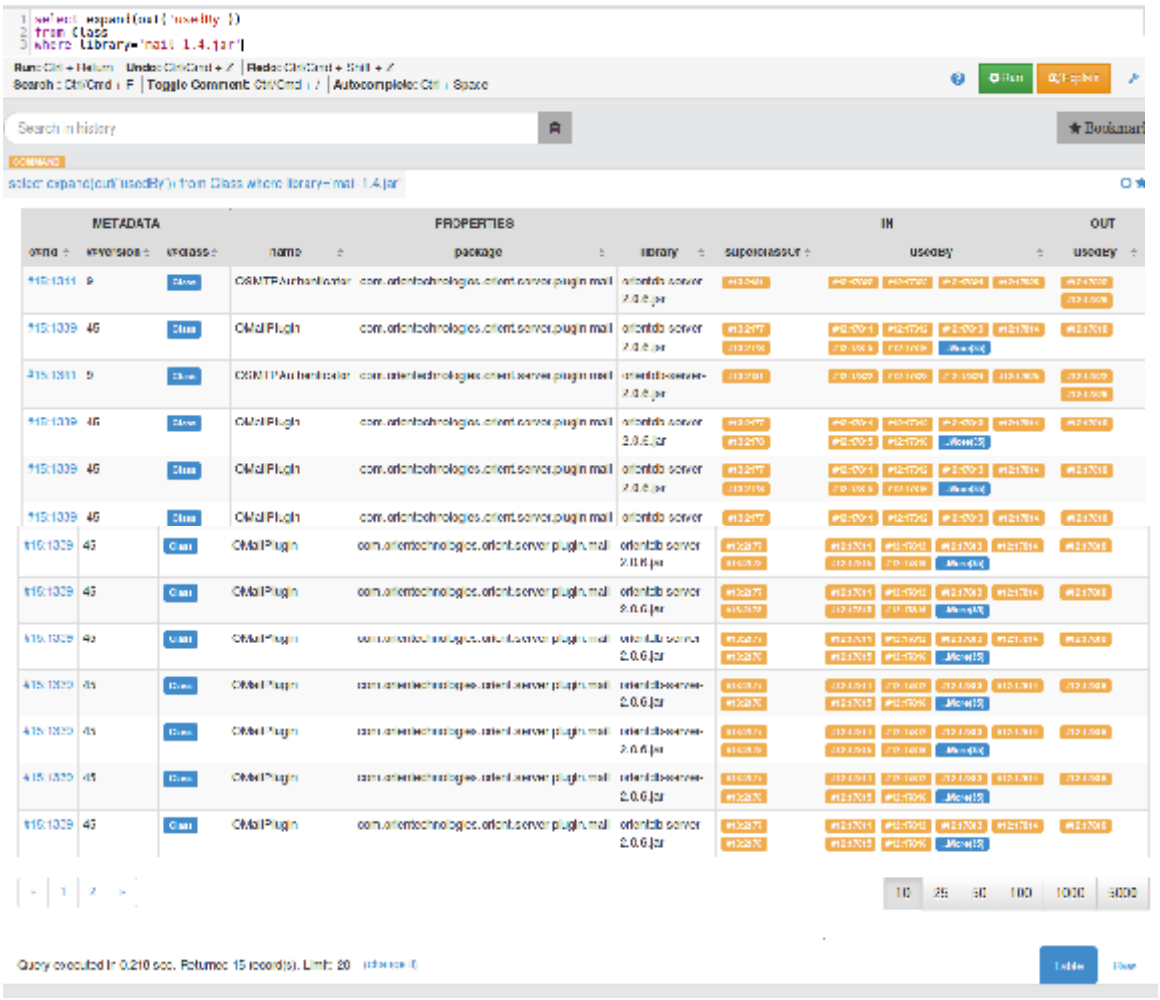
Step 2- Run different queries and note the time taken to execute the queries in browse editor.

Query 4 Searching the classes of OrientDB library with more relationships.

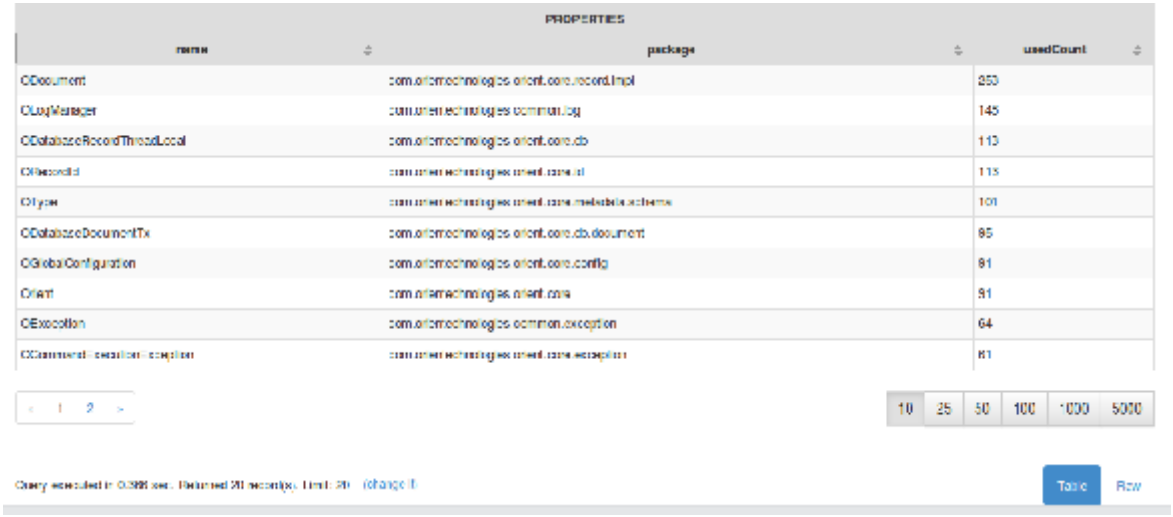
```
Select name,package,out('usedBy').size() as usedCount from Class where package like'com.oorientechnologies%' order by usedCount desc
```



Here if change is done to the class ODocument, it will affect 253 classes



Above snapshot helps to note down the response time of the query which is run using OrientDB java API.



PROPERTIES		
name	package	usedCount
ODocument	com.orientechnologies.orient.core.record.impl	253
OLogManager	com.orientechnologies.common.log	145
ODatabaseRecordThreadLocal	com.orientechnologies.orient.core.db	113
ORecordId	com.orientechnologies.orient.core.id	113
OType	com.orientechnologies.orient.core.metadata.schema	101
ODatabaseDocumentTx	com.orientechnologies.orient.core.db.document	95
OGlobalConfiguration	com.orientechnologies.orient.core.config	91
ORID	com.orientechnologies.orient.core	91
OException	com.orientechnologies.common.exception	64
OConcurrentModificationException	com.orientechnologies.orient.core.exception	61

From this a graph is created that contains a class diagram using the Java API OrientDB. The JAVA APIs are very simple to use, allow to model and populate a graph without need

to know SQL commands. This graph will be useful to collect information about class relationships. Their time to perform the query is noted down to analyze the performance.

6.2 Movie recommendation system with OrientDB

A movie recommendation engine is created using OrientDB. Download the data from online grouplens website. I have downloaded the data with 10,000 objects. The dataset contains data about users and their occupation, movies and their ratings. With the help of this dataset a graph schema will be generated in OrientDB to create a recommendation engine. In fig 6.1 classes with their attributes are depicted.

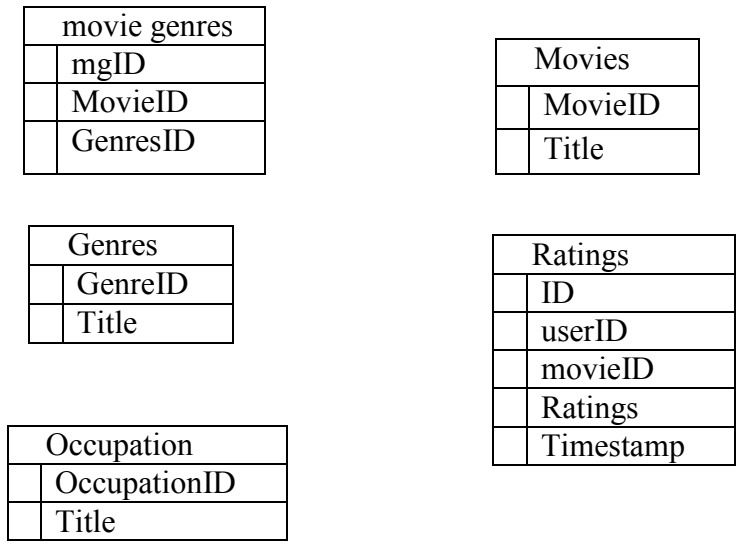
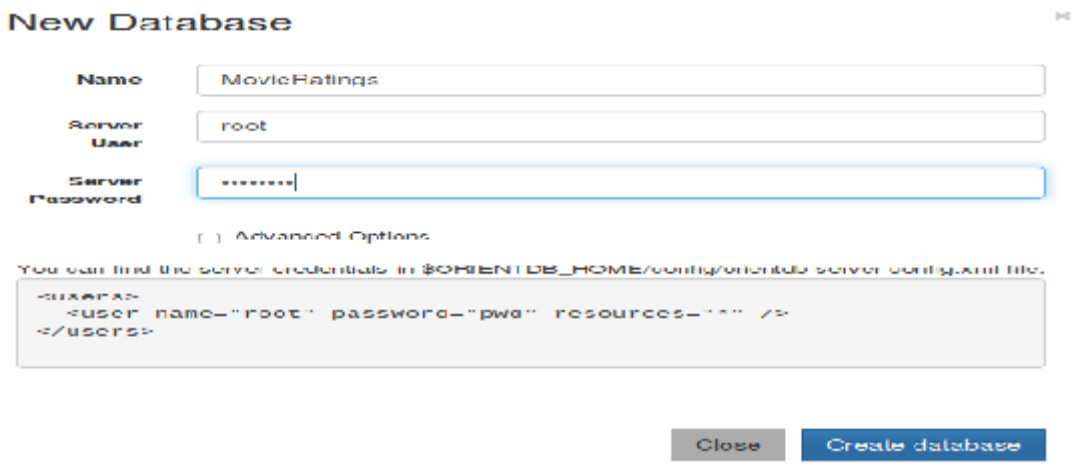
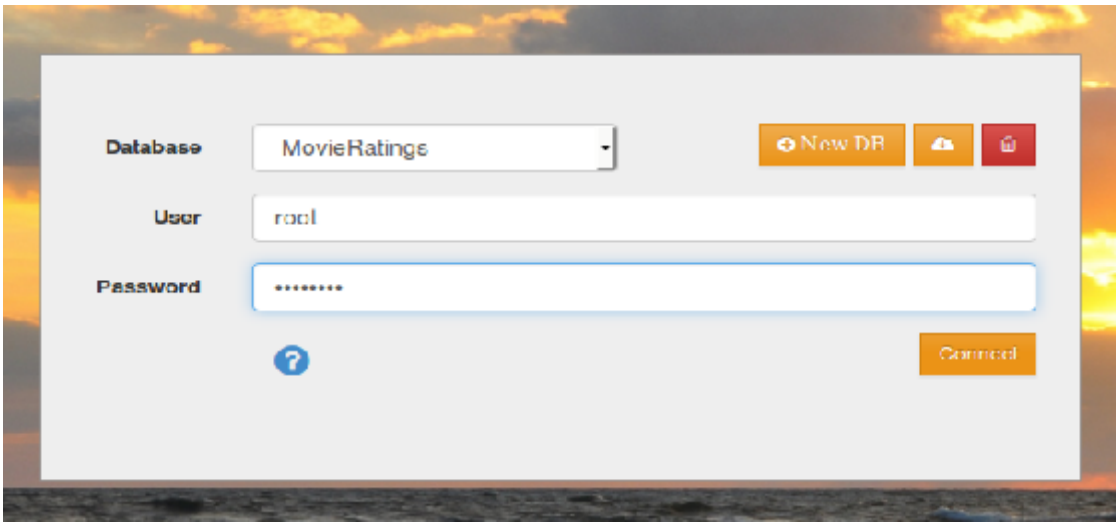


Fig 6.1 RDBMS Model

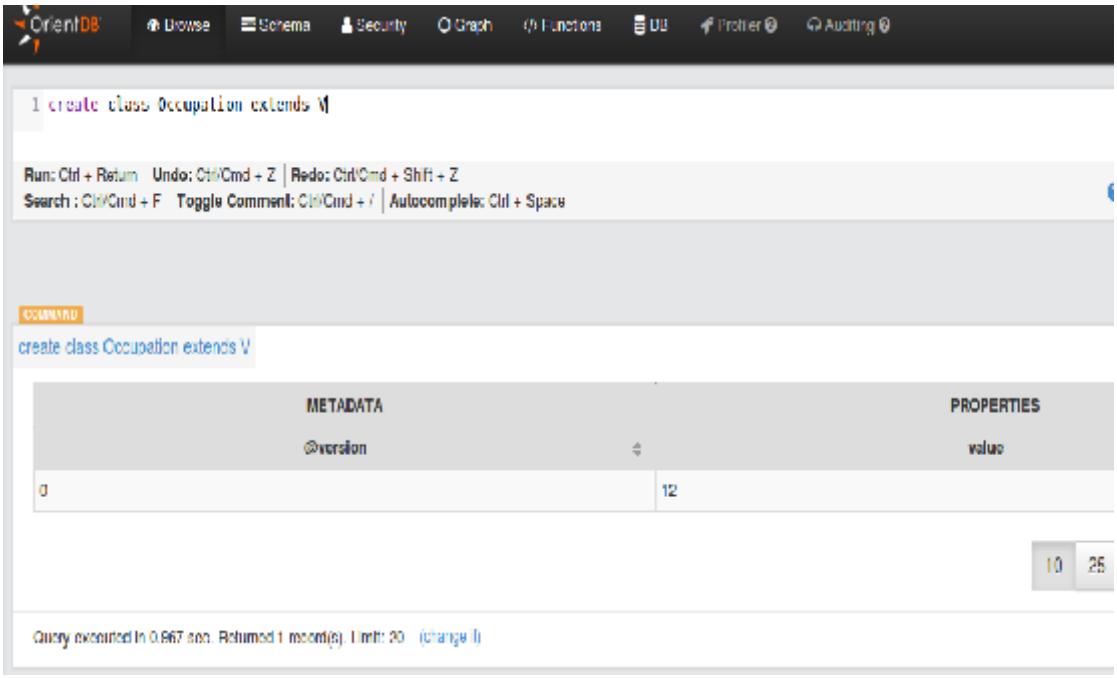
Steps to make a graph schema in OrientDB

Step1: Run OrientDB database and create MovieRatings database.



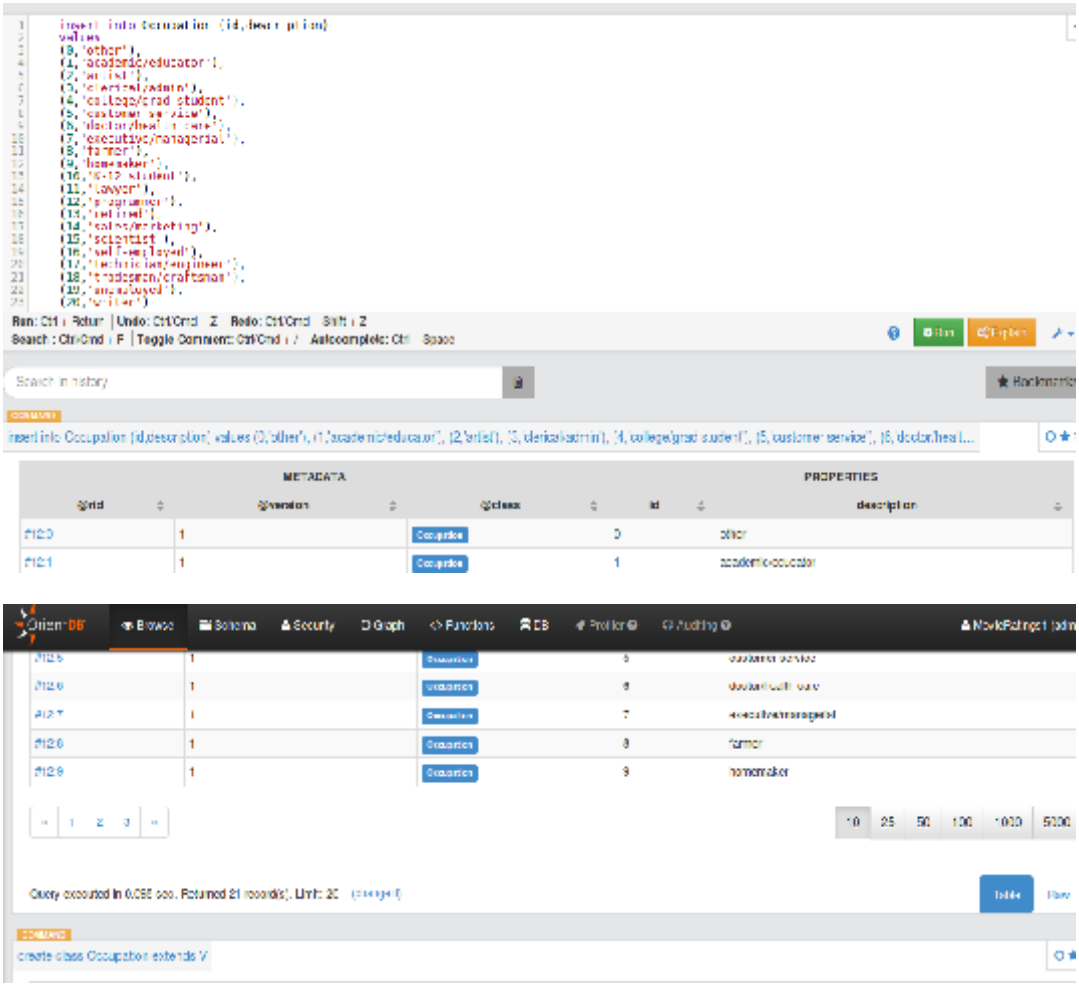


Step2: Create and populate the Occupation class that stores data about user's occupation.

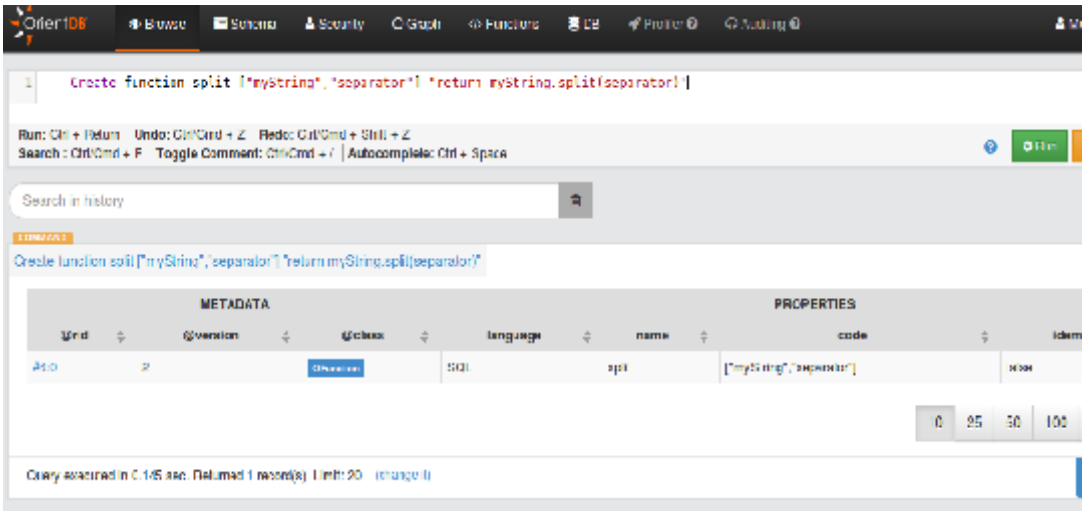


In above snapshot a query create class occupation extends is run without using java API, this query is run directly from OrientDB and its response time is noted. Earlier the queries were run using Java API with OrientDB. To make the comparison of queries when run without Java API and with Java API in OrientDB a strong conclusion can be withdrawn related to the retrieval (response) time of the queries.

Step3 : Write Data to be inserted in insertion script



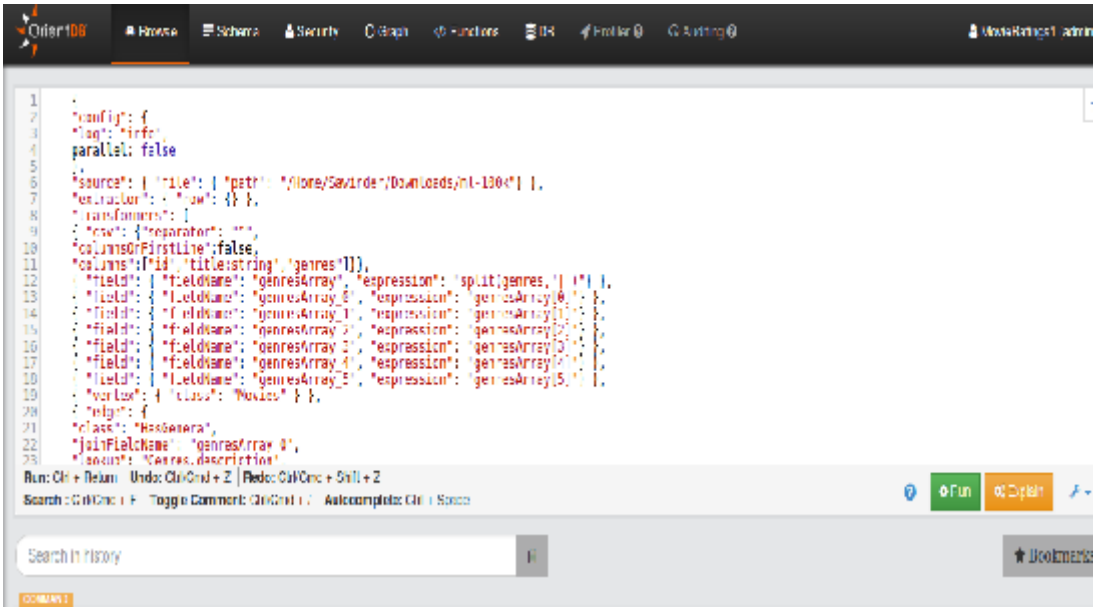
Step3: Create the function from OrientDB studio.



With its help data is imported with the help of OrientDB ETL. The Extractor Transformer and Loader(ETL) is a module for OrientDB provides support for moving data to and from OrientDB databases using ETL processes.

- Configuration-The ETL module uses a configuration file, written in JSON.
- Extractor-Pulls data from the source database.
- Transformers-Convert the data in the pipeline from its source format to one accessible to the target database.
- Loader-loads the data into the target database.

Step 4: The JSON file for OrientDB ETL allows the creation of both Movies and Genres nodes together with the edges between them.

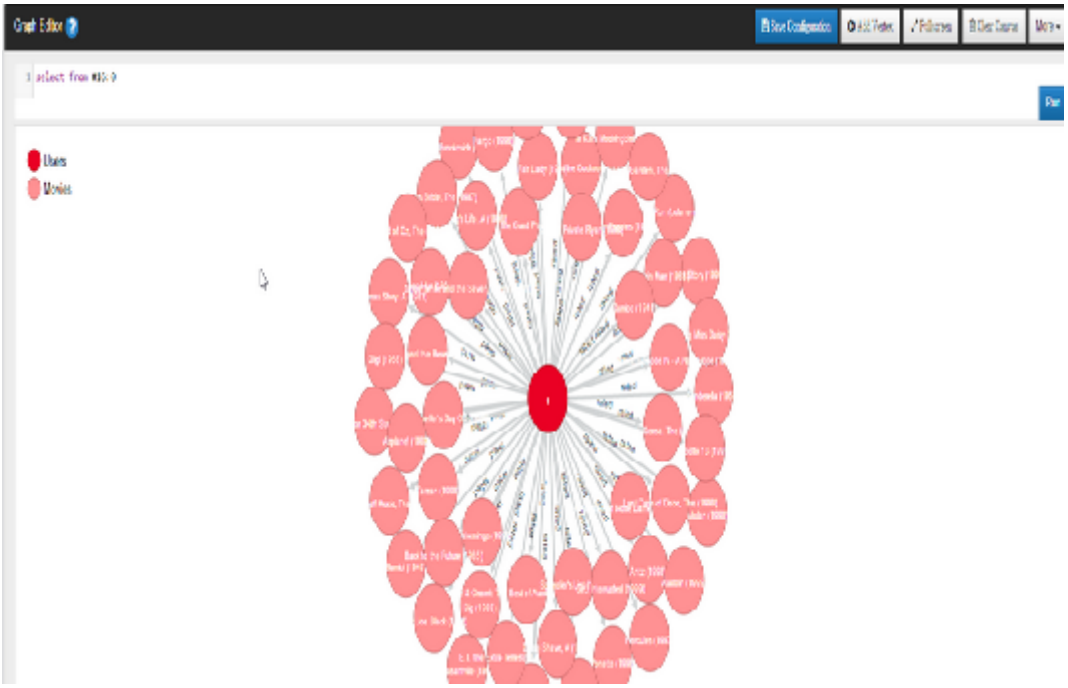


Now execute OrientDB ETL in OrientDB console.

In above snapshot a complete movie structure is seen. Relation between three different classes (Comedy, Animation, Children) to the story is shown in the graph.

Step 5: Import the user data into users class by executing this json file in OrientDB ETL and execute the script to load rating edges.

Step 6: Database stores Users class into the cluster 16 and Movies into the cluster 13, execute the query SELECT FROM #16:0, where #16:0 is Toy Story movie.



Step7: Execute different queries to obtain the result.

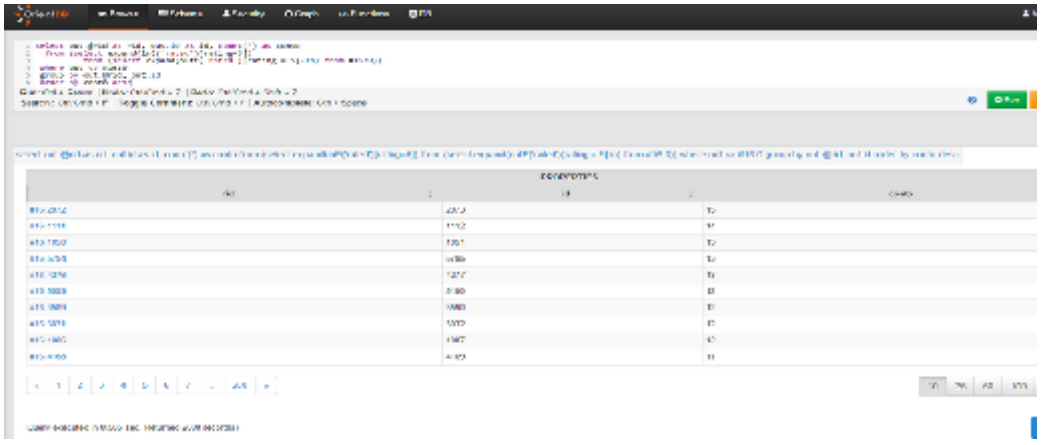
After insertion and execution of movie recommendation database, a graph is represented in OrientDB. Further, changes to this graph will be done by executing the queries so that a conclusion can be withdrawn i.e any changes made to one node in the class diagram will not affect the other node.

Query1- To take the one that has received more five stars between all the users.

```
Select @cid,title, int('rated')[ratings=5].size() as 5starNumber from (select expand(out('rated')[ratings=5].in) from #16:0) order by 5starNumber desc
```

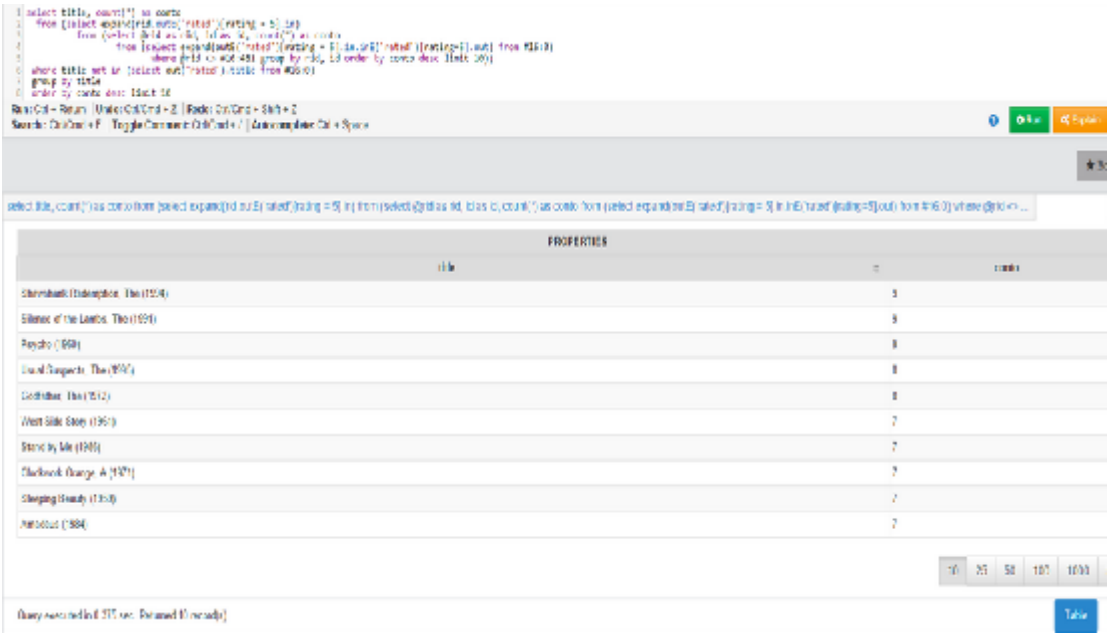
Query2- Search the users that gave max 5 stars to the films to which the users #16:0 has given stars.

```
Select out.@rid as rid, out.id as id,count(*) as conto from (select expand (int('rated') [ratings=5]) from (select expand(out('rated')[rating=5].in) from #16:0)) where out,,,. #16:0 group by out.@rid,out.id order by conto desc
```



Query3- Among the first 10 users similar to the user 16:0, which film has received more 5 stars and is still not present in the films rated by 16:0.

```
Select title,count(*) as conto from(select expand(rid.out('rated')[rating=5].in) from (select @rid as rid, id as id, count(*) as conto from (select expand(out('rated')[rating=5].in.int('rated')[rarings=5].out)from #16:0) where @rid <> #16:481 group by conto desc limit10)) where title not in (select out('rated').title from # 16:0) group by title order by conto desc limit 10
```



With the help of these three queries the response time is noted down and further analysis of performance will be done based on it. Also by running these queries we get to know how many records can be retrieved for that query execution time.

6.3 Performance Analysis

To analyse the time taken by three different queries a performance analysis is done in which response time to retrieve the query is noted down and according to the readings bar graph is generated. In fig 6.2 Time related bar graph is made to know how much time is taken by the queries when queries are run on OrientDB without any integration with the Java API. So, the time taken by these queries in comparison to the queries which are run with integration with java API is less.

In movie recommended system three queries have been executed and vertically execution time is drawn in the bar graph. Query 1 tells to take the one that has received more five stars between all the users. Query 2 searches those users that gave max 5 stars to

the films to which the users #16:0 has given stars. Query 3 tells that among the first 10 users similar to the user 16:0, which film has received more 5 stars and is still not present in the films rated by 16:0.

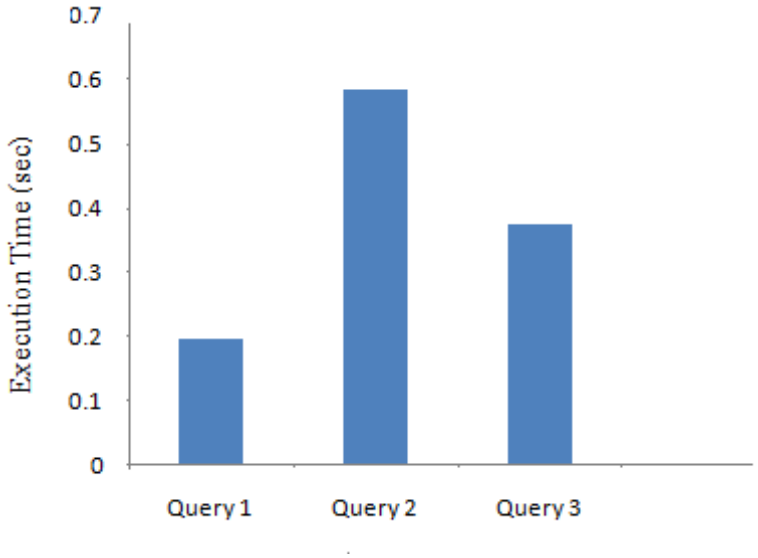


Fig 6.2 Time taken by queries during execution

In fig 6.2 query 1 which takes out the one that has received more five stars between all the users, takes 0.196 seconds of execution time and returns 18 records after comparison. Query 2 which search those users that gave max 5 stars to the films to which the users #16:0 has given stars, takes 0.585 seconds of execution time and returns 2000 records. Query 3 tells that among the first 10 users similar to the user 16:0, which film has received more 5 stars and is still not present in the films rated by 16:0, takes 0.376 seconds and returns 10 records. From the above bar graph it can be concluded that query 1 has only single nested query so takes lesser time to execute whereas query 3 has multiple nested loops so execution time to retrieve the data for query 3 increases. In query 2 the records to be retrieved are very large so it takes few milliseconds more that other two queries.

In this chapter, different queries are executed and their response time is noted down to analyse the performance of different queries and this helps to make easier conclusion why the response time of the queries vary. Class diagram with the integration of OrientDB and java API is depicted in the form of graph.

CHAPTER 7

Conclusion and Future Scope

NoSQL is a product which helps to handle issues like performance, scalability and complexity. Many enhancements have been provided by Non-relational databases over traditional relational databases such as cloud instances or commodity servers which helps to work on remote servers for storing, retrieving, updating the data through use of internet, not remaining strict to rigid schema for inserting data and hence it becomes easier to capture different varieties of data without making many changes at schema level.

NoSQL databases and OrientDB 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 will help to make it easier in representing a class diagram in graphical format. Till date there is no publication that has explained class diagram using OrientDB and querying in it to retrieve and update the class diagram without affecting the other classes. A case-study related to movie recommended system have been explained where classes are represented through nodes in OrientDB and relation between the nodes is generated so that it becomes easier for the user to understand a class diagram. With the help of queries data relationship between classes have been depicted through graph in OrientDB and their performance have been noted down.

In future further emphasis will be put on how to retrieve data with better response time. When class diagram is changed how the rest of the diagrams automatically will be changed corresponding to changed class diagram. We intend to extend our work to include other graph datastores initially, then exploring other datastore classes and storage models using the same methodology. As now the process works on given application and any change in datastore reflects the change in graphical format of OrientDB, further a change in corresponding document of that datastore will be made visible on OrientDB. It is a long way towards integrating heterogeneous data models, combining their potentials, and polyglot persistence.

REFERENCES

[1] Kaisler, S., Armour, F., Espinosa, J.A. and Money, W., 2013, January. Big data: issues and challenges moving forward. In System Sciences (HICSS), 2013 46th Hawaii International Conference on (pp. 995-1004). IEEE.

[2] Snijders, C., Matzat, U. and Reips, U.D., 2012. " Big Data": big gaps of knowledge in the field of internet science. International Journal of Internet Science, 7(1), pp.1-5.

[3] Leavitt, N., 2010. Will NoSQL databases live up to their promise?. *Computer*,*43*(2), pp.12-14.

[4] Robinson, I., Webber, J. and Eifrem, E., 2015. Graph Databases: New Opportunities for Connected Data. " O'Reilly Media, Inc."

[5] Cook, D.J. and Holder, L.B. eds., 2006. Mining graph data. John Wiley & Sons.

[6] Chen, C.P. and Zhang, C.Y., 2014. Data-intensive applications, challenges, techniques and technologies: A survey on Big Data. Information Sciences,275, pp.314-347.

[7] Putnik, G., Sluga, A., ElMaraghy, H., Teti, R., Koren, Y., Tolio, T. and Hon, B., 2013. Scalability in manufacturing systems design and operation: State-of-the-art and future developments roadmap. CIRP Annals-Manufacturing Technology, 62(2), pp.751-774.

[8] Sakr, S., Liu, A., Batista, D.M. and Alomari, M., 2011. A survey of large scale data management approaches in cloud environments. IEEE Communications Surveys & Tutorials, 13(3), pp.311-336.

[9] Padhye, V. and Tripathi, A., 2015. Scalable transaction management with snapshot isolation for NoSQL data storage systems. IEEE Transactions on Services Computing, 8(1), pp.121-135.

[10] Subramaniyaswamy, V., Vijayakumar, V., Logesh, R. and Indragandhi, V., 2015. Unstructured data analysis on big data using map reduce. Procedia Computer Science, 50, pp.456-465.

[11] SIDDIQA, A., KARIM, A. and Abdullah, G.A.N.I., 2016. Big data storage technologies: a survey. Frontiers, 1.

[12] Diack, B.W., Ndiaye, S. and Slimani, Y., 2013. CAP Theorem between Claims and Misunderstandings: What is to be Sacrificed?. International Journal of Advanced Science and Technology, 56, pp.1-12.

[13] Xiao, Z. and Liu, Y., 2011, June. Remote sensing image database based on NOSQL database. In *Geoinformatics, 2011 19th International Conference on*(pp. 1-5). IEEE.

[14] Vicknair, C., Macias, M., Zhao, Z., Nan, X., Chen, Y. and Wilkins, D., 2010, April. A comparison of a graph database and a relational database: a data provenance perspective. In Proceedings of the 48th annual Southeast regional conference (p. 42). ACM.

[15] Zhang, X. and Xu, F., 2013, September. Survey of research on big data storage. In Distributed Computing and Applications to Business, Engineering & Science (DCABES), 2013 12th International Symposium on (pp. 76-80). IEEE.

[16] Tudorica, B.G. and Bucur, C., 2011, June. A comparison between several NoSQL databases with comments and notes. In 2011 RoEduNet International Conference 10th Edition: Networking in Education and Research (pp. 1-5). IEEE.

[17] Grolinger, K., Higashino, W.A., Tiwari, A. and Capretz, M.A., 2013. Data management in cloud environments: NoSQL and NewSQL data stores.Journal of Cloud Computing: Advances, Systems and Applications, 2(1), p.1.

[18] Hajoui, O., Dehbi, R., Talea, M. and Batouta, Z.I., 2015. An advanced comparative study of the most promising NoSQL and NEWSQL databases with a multi criteria analysis method.Journal of Theoretical and Applied Information Technology, 81(3), p.579.

[19] Angles, R. and Gutierrez, C., 2008. Survey of graph database models. ACM Computing Surveys (CSUR), 40(1), p.1.

[20] Tesoriero, C., 2013. Getting Started with OrientDB. Packt Publishing Ltd.

[21] Tudorica, B.G. and Bucur, C., 2011, June. A comparison between several NoSQL databases with comments and notes. In 2011 RoEduNet International Conference 10th Edition: Networking in Education and Research (pp. 1-5). IEEE.

[22] Li, Y. and Manoharan, S., 2013, August. A performance comparison of SQL and NoSQL databases. In Communications, Computers and Signal Processing (PACRIM), 2013 IEEE Pacific Rim Conference on (pp. 15-19). IEEE.

[23] Ceri, S., Gottlob, G. and Tanca, L., 1989. What you always wanted to know about Datalog (and never dared to ask). IEEE Transactions on Knowledge and Data Engineering, 1(1), pp.146-166.

[24] Keller, A.M., Jensen, R. and Agarwal, S., 1993, June. Persistence software: Bridging object-oriented programming and relational databases. In ACM SIGMOD Record (Vol. 22, No. 2, pp. 523-528). ACM.

[25] Pokorny, J., 2013. NoSQL databases: a step to database scalability in web environment. International Journal of Web Information Systems, 9(1), pp.69-82.

[26] Armbrust, M., Lanham, N., Tu, S., Fox, A., Franklin, M.J. and Patterson, D.A., 2010, June. The case for PIQL: a performance insightful query language. In Proceedings of the 1st ACM symposium on Cloud computing(pp. 131-136). ACM.

[27] Vicknair, C., Macias, M., Zhao, Z., Nan, X., Chen, Y. and Wilkins, D., 2010, April. A comparison of a graph database and a relational database: a data provenance perspective. In Proceedings of the 48th annual Southeast regional conference (p. 42). ACM.

[28] Batra, S. and Tyagi, C., 2012. Comparative analysis of relational and graph databases. International Journal of Soft Computing and Engineering (IJSCE),2(2), pp.509-512.

[29] Buerli, M. and Obispo, C.P.S.L., 2012. The current state of graph databases.Department of Computer Science, Cal Poly San Luis Obispo, mbuerli@ calpoly. edu, pp.1-7

[30] Kaur, K. and Rani, R., 2013, October. Modeling and querying data in NoSQL databases. In Big Data, 2013 IEEE International Conference on (pp. 1-7). IEEE.

[31] Moniruzzaman, A.B.M. and Hossain, S.A., 2013. Nosql database: New era of databases for big data analytics-classification, characteristics and comparison. arXiv preprint arXiv:1307.0191.

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

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

[34] 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/

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

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

[37]] 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

List of Publications

Sawinder kaur and Karamjit Kaur.Visualizing class diagram in OrientDB using NoSQL Data Store, In 2016 IJISET International Journal of Innovative Science Engineering and Technology (Published). IJISET, June 20, Volume 3, Issue 6.

Youtube Link

https://youtu.be/as83QQ7Bqk4