ORIENT DB – A MULTI MODEL NOSQL DATABASE

Naveena Meka
Department of Computer science
University of Bridgeport

Udayasreereddy Teegulla
Department of Computer Science
University of Bridgeport

Abstract: In this paper, we discuss about Orient DB database and its implementation, there by highlighting its features. Orient DB is a multi-model database, so it supports all kinds of models. We used graph data model as it is well known for it and an openbeer data set was used in order to highlight its main feature, which is to manage the relationships using direct links between the records. A user-friendly front end was created for the open beer data, where we can clearly identify all the relations present in the data.

1. INTRODUCTION

In the recent times, the term NOSQL databases has been talked about everywhere, there has been a wide range of growth in NOSQL products and solutions. Considering the problems of the relational databases such as Performance, Scalability, ability to handle internal failures and inflexible schema. The main reason for these types of problems is large amounts of data, when the relational databases were first developed their main focus was not on handling large quantities of data. Now in the present scenario considering the large number of internet users, there is a heavy need for the type of databases that can handle Big Data[1].

The most common models for handling the problems faced in todays scenario are Key-value, Column- oriented, Document and Graph Databases. In Key- value type of model the data is stored in multiple servers in the form of Key-value pairs. Column-based data models store the data in the form of columns, where it becomes easy to search the data as they are divided nicely. In case of the data which is not ordered that does not contain any structure, Document data modelling is used. In case of Graph data modelling the vertices are connected using edges to form a well organised graph structure. Some of the well-known databases for graph data modelling are neo4j and Orient DB. However, all of these data models have their own advantages and disadvantages according to the requirements of the user. What makes Orient DB interesting when compared to other databases is its ability to act as a multi model, where it supports all four categories of the model.

This paper is structured as follows. Section 2 discusses about related work, focussing on the reason for choosing this database. Section 3 explains the data model in detail. Section 4 provides the details about the data set that has been used. In the Section 5 all the Implementation details from Installation to execution has been discussed.

2. RELATED WORK

The need for Graph Databases was started during mid 1960's when existing databases such as IBM'S IMS strict tree structure, called as Hierarchical modelling started interfering with virtual records. In order to address such problem Network model databases were developed and around 1980's Graph data modelling was started. Around 20th Century, many Databases such as Neo4j, Arango DB, Orient DB, Mark Logic came into existence by supporting the ACID properties. Allegro Graph was among the initial graph databases that was implemented using C, C# and Java languages were Resource Distribution Framework was used. Similarly, Amazon Neptune was developed by Amazon, it is a part of Amazon Web service and it is used by the users as a web-service. However, many implementation details of this database have not been disclosed. Arango DB is a native database which supports three different types of data models and uses its own query language called AQL for using it. Even though it supports different data models and gives fast results, Arango DB's cluster is limited by the available resources of CPU, memory, disk and bandwidth latency. Neo4j is another open source NoSQL database which uses Graph data modelling which provides high clustering and a web administration tool. But when compared to other Databases, Orient DB is faster in performance, it is very easy to use as it uses SQL for querying and it is very famous for its graph data model. In fact it is proved in a IBM research institute that Orient DB[6] is ten times faster than neo4j in different scenarios of Workloads.

3. DATA MODEL

Graph Data model consists of two main components which are vertices and edges. A vertex is nothing but a node which is used to store the data and edge represents the relationship between the vertices. Using Graph Data

model we can increase the flexibility to our data, as unlike the traditional databases we don't use join operations or foreign keys. Instead, all the relationships are stored within the vertices. Particularly, in case of Orient DB the relations are stored within the vertices, that is as documents [3].

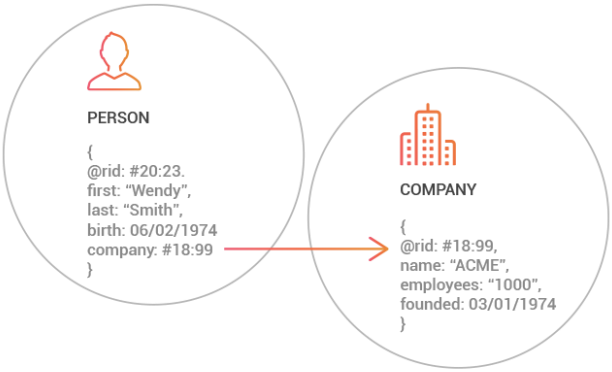

Fig.1: A direct link between two vertices.

As we can see from Fig.1, instead of using JOIN operation or any key to access the company details a person is working, a direct link is established between them. By doing so, Performance will be increased very rapidly. Even we consider the real-world scenario, many social media applications, medical applications and analytic applications use graph data modelling as they mostly deal with connected data.

4. DATA SET

An openbeer data set has been used for implementing our work. This Openbeer dataset has been directly downloaded from Kaggle and was pre-processed to make it compatible for the Graph Database. This Openbeer data set contains information about different kinds of beers, the category it belongs to, its brewery and its Style. This Dataset consists around 6000 rows in Beer vertex, 11 in Category vertex, 142 in Style vertex and 1500 in Breweries.

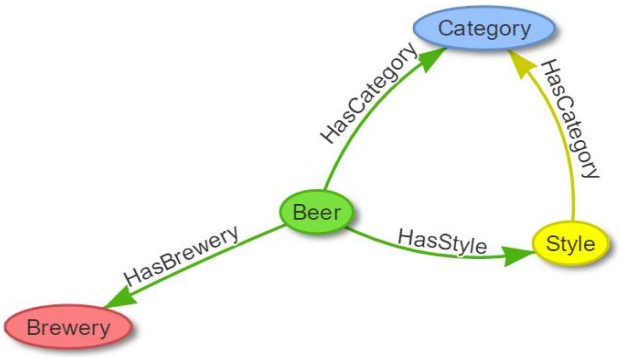

Fig.2: Graph Model for the Openbeer data set.

From Fig.2, we can easily identify the Vertices and the relationships present in the dataset. Beer, Category, Style and Brewery are the Vertices and the relationships between them are HasBrewery, HasStyle and HasCategory. The main reason for choosing this dataset is to highlight the feature of the Orient DB's Graph Data modelling where it displays all the relationships present for the node in a feasible manner.

5. IMPLEMENTATION

5.1 Installations and Connection Establishment

Orient DB database is available as both community (Open source) and Enterprise Edition. It can run on any Operating System as long as it contains Java Virtual Machine [4]. The Following are the steps for installing and running the Orient DB:

- Download and install oracle JDK 1.7 and it also requires Java version to be 1.7 or higher.
- Download Orient DB from the website.
- Extracting the Orient DB into desired location.
- Starting the server by running the commands server.bat or server.sh on windows or Mac respectively.
- After the Orient DB gets started open the URL http://localhost:2480 in the browser window.

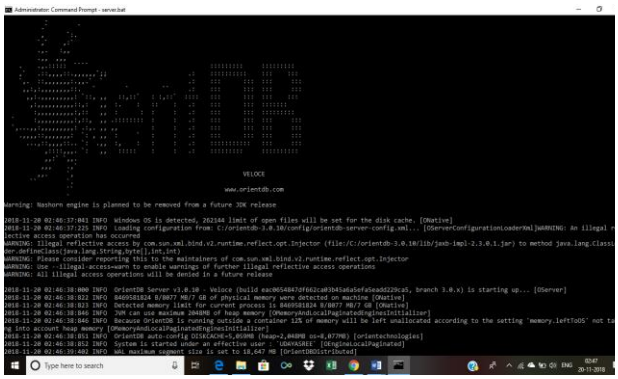

Fig.3: Starting the Orient DB

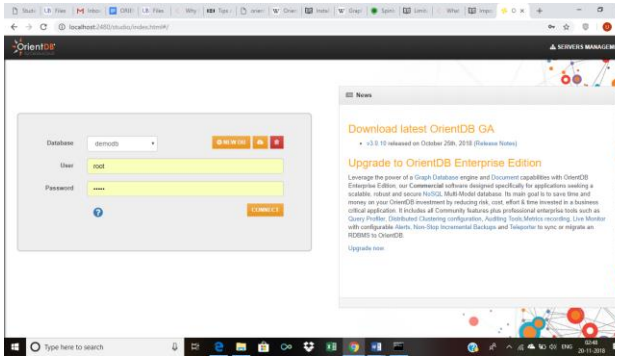

Fig.4: Connecting to the Orient Db server

The above Fig.3 and Fig.4 show how to activate OrientDB server and to connect to OrientDB.

5.2 Importing the Data Set

Orient DB provides ETL process for transporting the data to and from the Database.This ETL process consists of Extractor, Transformer and Loader. Extractor is used to fetch the data from the source and in order to convert the data into the suitable format transformer is used.The work of the Loader is to just the load the transformed data into the database. In order to execute this process all ETL work is scripted in a Configuration file, which is written in JSON [4].

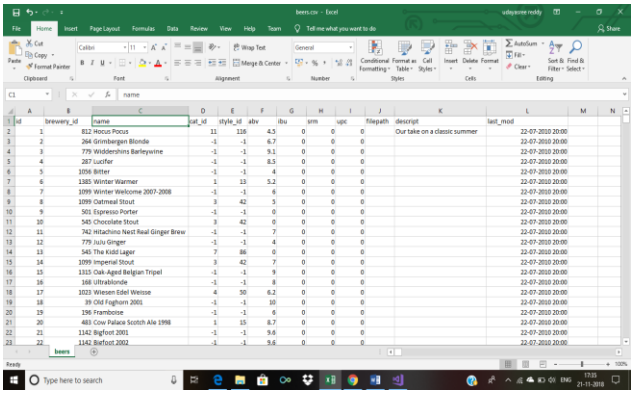

Fig.5: Beer.csv file that provides all information about Beers.

Now, for the above Beer table data set as shown in Fig.5, we write a Configuration file for loading the data into Database.

JSON FILE

```
{
  "config" : { "haltOnError": false },
  "source": { "file": { "path": "C:\orientdb-
3.0.10\bin\openbeerdb_csv\beers.csv" } },
  "extractor": { "csv": { "columns":
["id","brewery_id","name","cat_id","style_id"
,"abv","ibu","srm","upc","filepath","descript
","last_mod"],

"columnsOnFirstLine": true } },
  "transformers": [
    { "vertex": { "class": "Beer" } },
    { "edge": { "class": "HasCategory",
"joinFieldName": "cat_id", "lookup":
"Category.id" } },
    { "edge": { "class": "HasBrewery",
"joinFieldName": "brewery_id", "lookup":
"Brewery.id" } },
    { "edge": { "class": "HasStyle",
"joinFieldName": "style_id", "lookup":
"Style.id" } }
  ],
  "loader": {
    "orientdb": {
      "dbURL":
"plocal:../databases/openbeerdb",
      "dbType": "graph",
      "classes": [
  {"name": "Beer", "extends": "V"},
  {"name": "HasCategory", "extends": "E"},
  {"name": "HasStyle", "extends": "E"},
  {"name": "HasBrewery", "extends": "E"} ],
"indexes": [
```

```
{"class":"Beer", "fields":["id:integer"],
"type":"UNIQUE" } ] } }
```

In Fig.6, Oetl.bat is the command that is used for running this file in case of Windows and for Mac the command is oetl.sh.

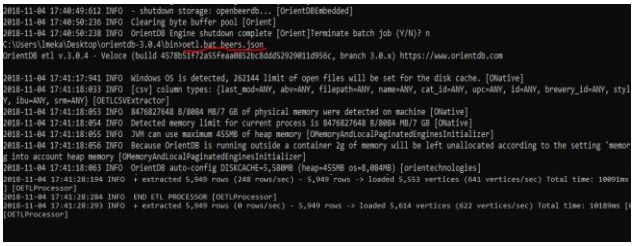

Fig.6: Running beer.json file

In the similar fashion all the four vertices like beer, breweries, Style and Category has been imported into the Database.

5.3 Data modeling using Graph Data model in Orient DB

In order to model any kind of data, identifying its tables and relationships is a very important step and the below diagram Fig.7 explains all the relations present in the open beer dataset in a visualized manner.

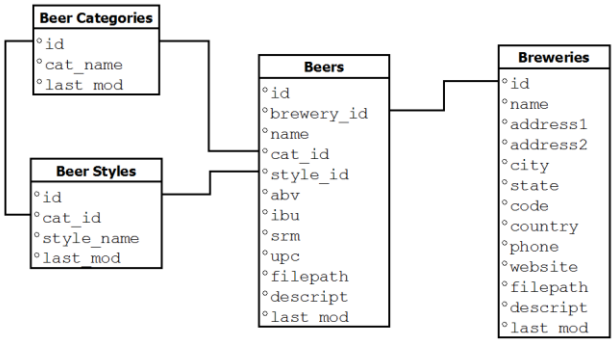

Fig.7: ER Diagram of Open Beer Data set.

Once we load the data into Database for all the four vertices, the information gets displayed in the local host UI of the orient DB. Creating a particular can be done in a json file as shown above or we can also create a particular vertex class and then create vertex in that class [5].

Orientdb>CREATE VERTEX Beer EXTENDS V
Orientdb>CREATE VERTEX Beer

Once, we import all the vertex classes brewery, Styles and Categories we will be able to see all of them in the Orient DB's UI as shown in Fig.8.

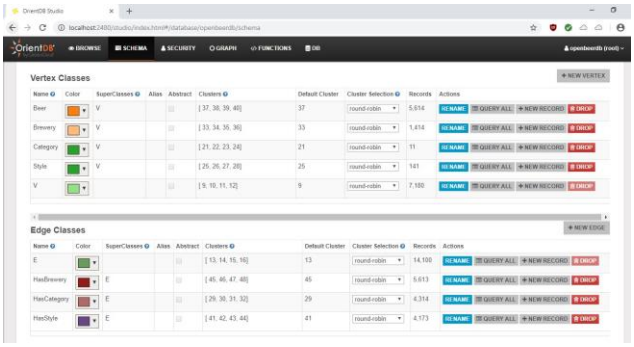

Fig.8: Vertex Classes and Edges of the Data set

5.4 Performing Queries

Performing Queries in the Orient DB is very simple as it can be done using SQL language. The basic SQL syntaxes including functions, procedures, aggregation operators are same as SQL [2]. But, what makes the Orient DB special is its ability to show all the relations that are present to a node, even if we perform a simple query. In Fig.9, we can clearly see that after executing a simple query, if we click on a particular node it shows all the relations and modularity's of that particular node.

Once, the user selects a particular beer, he will be able to see all styles and the breweries of that particular beer.

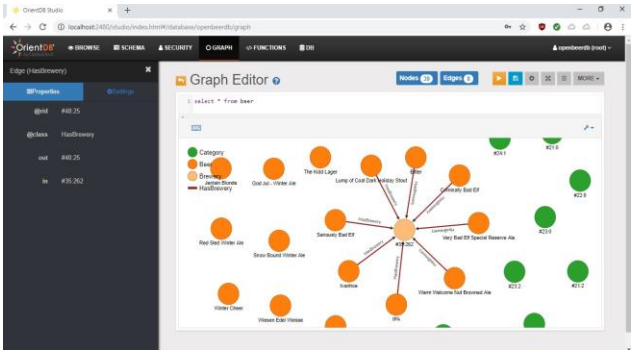

Fig.9: Projection of all the beers that have been brewed from brewery no 48.25.

5.5 Application Development

A front-end application has been developed for the work that we have done using .net, so as to demonstrate it in a feasible way. The application is built in such a way, where once you start it; it shows all the categories that are present in the database, which is similar to the query that has been performed in Fig.9. For the later part, the following query has been used.

```
SELECT EXPAND( BOTH() ) FROM Beer WHERE
name = 'Petrus Dubbel Bruin Ale'
```

From Fig.10, we can see that once user selects a particular beer, he will be able to see all styles and the breweries of that particular beer.

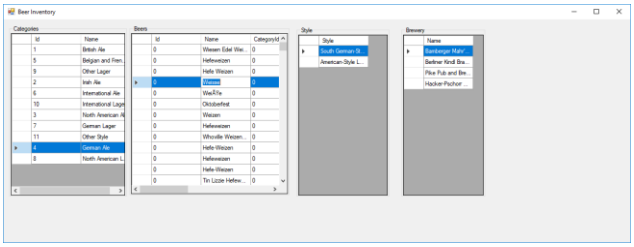

Fig.10: Screenshot of the front-end interface.

6. CONCLUSION

Orient DB is a simple database to work, with the additional advantage of it to be able to work with different kinds of Data models. Even though, Orient DB is a multi-model database, it is well known for its graph interface. It is a very apt choice when you have a dataset that has complicated relations in them[7]. It is mostly used in case of scenarios where we need to analyze something from the existing Data set. From the above work, we can clearly understand that Orient DB gives high performance and flexibility when compared to other graph databases.

7. REFERENCES

[1] http://www.orientdb.com/docs/last/index.html
[2] https://orientdb.org/docs//3.0.x/gettingstarted/Tutorial-SQL.html
[3] https://orientdb.org/docs//3.0.x/datamodeling/Tutorial-Document-and-graph-model.html
[4] https://orientdb.com/docs/2.1.x/Configuration-File.html
[5] https://github.com/orientechnologies/orientdb
[6] https://db-engines.com/en/system/OrientDB
[7] https://users.dcc.uchile.cl/~cgutierr/papers/surveyGDB.pdf