Erasmus Mundus Master's Programme in Information Technologies for Business Intelligence (IT4BI) 2015-2017

GRAPH DATABASES AND ORIENTDB

INFO-H-415: Advanced Databases (Project)

Professor: Esteban Zimányi
Teaching Assistant: Stefan Eppe

Authors:
Ahsan Bilal
Madiha Khalid

Contents

Abstract

In recent years, more and more companies provide services that cannot be anymore achieved efficiently using relational databases. As such, these companies are forced to use alternative database models such as XML databases, object-oriented databases, and document-oriented databases and, more recently graph databases. Graph databases only exist for a few years.

In this document we are exploring OrientDB as Graph Database model as NoSQL database. The main goal of this project is to provide theoretical, technical details and debates on some powerful features of OrientDB. We provide some comparison attempts between OrientDB 2.1.8 and SQL Server 2012, they are mostly focused on MovieLens dataset and build recommendation engine.

Introduction to NoSQL

NoSQL is not about saying that SQL should never be used or SQL is dead, neither a negation of the traditional RDBMS ecosystem, it just stands for "Not only SQL".

NoSQL Definition: Next Generation Databases mostly addressing some of the points: being non-relational, distributed, open-source and horizontally scalable[1].

NoSQL databases have become the first alternative to relational databases, with scalability, availability, and fault tolerance being key deciding factors.

Why NoSQL?

The data landscape has changed. During the past 15 years, the explosion of the World Wide Web, social media, web forms you have to fill in, and greater connectivity to the Internet means that more than ever before a vast array of data is in use.

New and often crucial information is generated hourly, from simple tweets about what people have for dinner to critical medical notes by healthcare providers.

Systems designers no longer have the luxury of closeting themselves in a room for a couple of years designing systems to handle new data. Instead, they must quickly create systems that store data and make information readily available for

Industrial Revolution of Data

Ninety percent (90%) of all the data in the world has been generated over the last two years. Internet-based companies are awash with data that can be grouped and utilized.

BigData

BigData makes it possible to achieve research results that cover a wide range of issues, and can tell us a great deal about developments in the world in many different areas.

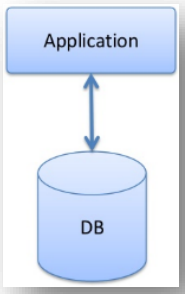
[1] *NOSQL*. NOSQL databases http://nosql-database.org/

search, consolidation, and analysis. All of this means that a particular kind of systems technology is needed.

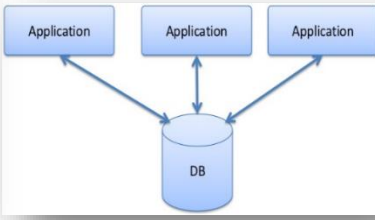
Let's have look at some driving trends which leads developers to think beyond the RDBMS structure.

1- Massively increase of Data Size.
2- Data Connectivity in application like social media.
3- Semi-structured Information.
4- Different architecture used to build application.

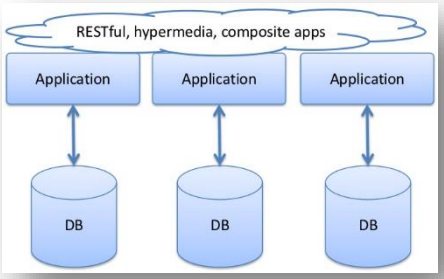
Single Application (1980's)



Integration Database Antipattern (1990's)



Service-Oriented Architecture (2000's)



The original intention of NoSQL approach has been creation of modern web-scale databases. NoSQL is designed for distributed data stores with needs of scaling of the data (e.g. Facebook or Twitter, which accumulates terabits of data every single day). The basic characteristic belong:

- schema-free
- easy replication support
- own API
- consistency (BASE/ *ACID* (Atomicity, Consistency, Isolation, Durability) transactions)
- huge amount of data
- unstructured data
- data on multiple servers in the cloud

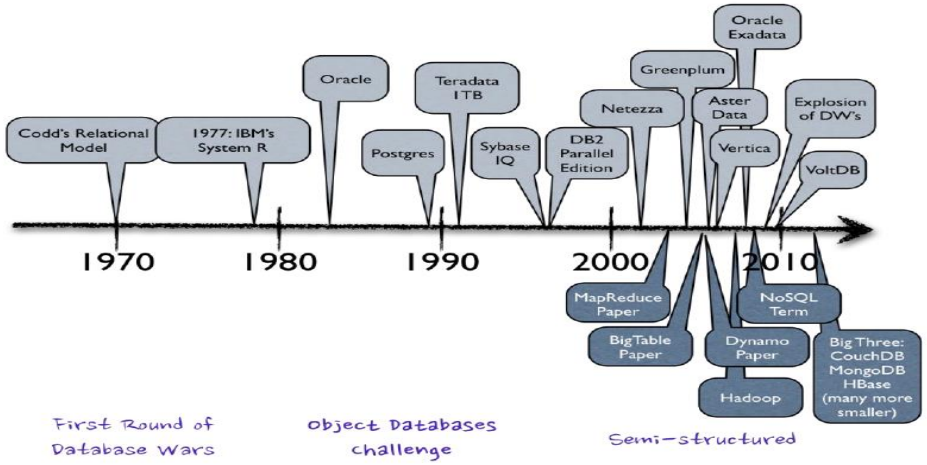


Figure 1: History of Database Evolution of NoSQL

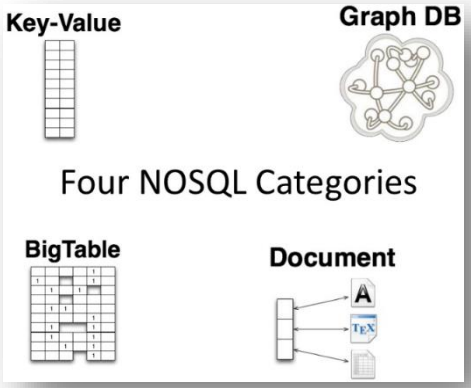
When to choose an RDBMS and NoSQL

	Pros	Cons
NoSQL	1. flexible data model without restrictions on data 2. suitability for running in the Cloud 3. good options for horizontal scaling without buying additional expensive hardware 4. suitability for storing of rapidly growing data 5. suitability for hierarchical, heavily interconnected or unstructured data 6. suitability for creation semantic model (semantic web)	1. unsuitability for users with small programming skills → difficulty to manage database and make database queries 2. partial instability of open source projects (the most of NoSQL projects are open source) → on-going development process → some required features could be missing 3. bigger difficulty to install and set-up than RDBMS
RDBMS	1. suitability for structured data with the ability to ask different questions all the time 2. native referential integrity and ACID transactions 3. well-known relational model which uses well-known query language (SQL)	1. unsuitability for storing application entities in a persistent and consistent way 2. unsuitability for hierarchical application objects with query capability into them 3. unsuitability for storing large trees or networks 4. unsuitability for running in the Cloud and usage as a distributed database 5. unsuitability for very fast growing data which is not possible to process on a single machine 6. not easy accessible horizontal scaling (without buying more expensive hardware) 7. performing JOIN operations

NoSQL Categories

There are following basic NoSQL categories:

a. Graph Databases
b. Key-Value
c. BigTable
d. Document



The right choice of database model for specific use case is very important and also difficult task. We can see comparison between relational and NoSQL databases according to the scaling size and database model complexity on Figure 2.

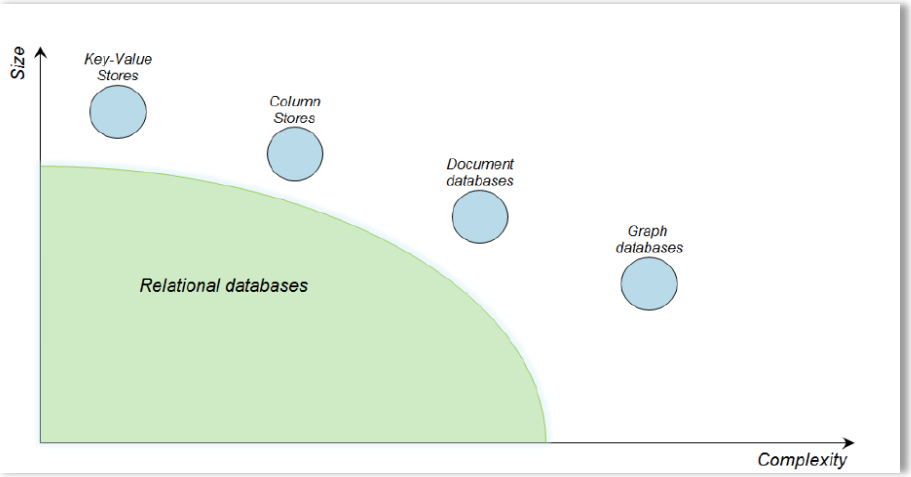


Figure 2: Positions of NoSQL databases (scaling vs. complexity)

Some of the NoSQL vendors are as shown in:

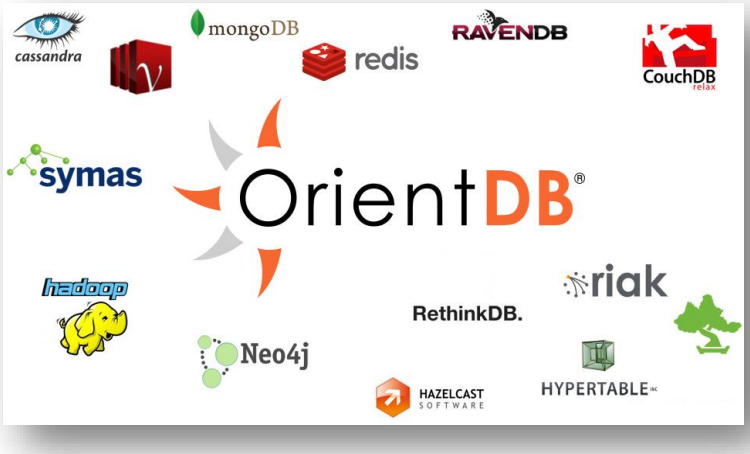


Figure 3: NoSQL Vendors

NoSQL Graph Databases

Graph Databases recently gained lot of attention due to its performance and features which, combined together, offer a tool that is by far different from any other product in the DBMS ecosystem.

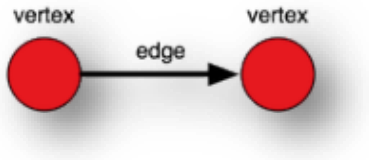
Graph databases are a rising tide in the world of big data insights, and the enterprises that tap into their power realize significant competitive advantages and can handle relationships in an easier and faster way compared to traditional databases.

Graph databases can be especially used when following characteristics are desirable:

- Develop application related with social networking
- To dynamically build relationships between objects that have dynamic properties
- To build database incrementally through programming
- To avoid very nested JOIN operations (thanks to fast navigation between graph entities)

Graph Structure

A graph (or network) is a data structure. It is composed of vertices (dots) and edges (lines). Many real-world scenarios can be modelled as a graph. This is not necessarily inherent to some objective nature of reality, but primarily predicated on the fact that humans subjectively interpret the world in terms of objects (vertices) and their respective relationships to one another (edges).



Graph Database Structure

The popular data model used in graph computing is the property graph. The following example demonstrate graph modelling via scenario.

TinkerPop Blueprints stack [2]

TinkerPop blueprints provide interfaces and implementations for the property graph data model under apache2 license. Technology stack contains:

- **Pipes:** data flow framework
- **Gremlin:** a graph traversal language
- **Frames:** an object-to-graph mapper
- **Rexter:** a graph server

Now TinkerPop3 made a sharp distinction between the various TinkerPop projects: Blueprints, Pipes, Gremlin, Frames, Furnace, and Rexster. With TinkerPop3, all of these projects have been merged and are generally known as Gremlin. **Blueprints** → Gremlin Structure API : **Pipes** → GraphTraversal : **Frames** → Traversal : **Furnace** → GraphComputer and VertexProgram : **Rexster** → GremlinServer.[3]

Gremlin Language

Gremlin is a graph manipulation language. It is specialized to work with Property graphs. Gremlin is a part of TinkerPop Blueprints stack. It provides support for Java and it supports multiple traversal patterns.

---

[2] Apache Tinkerpop3 https://tinkerpop.incubator.apache.org/
[3] Apache TinkerPop3 Documentaion http://tinkerpop.apache.org/docs/3.1.0-incubating/#_tinkerpop3

Gremlin main features are:

- manual working with graph (create, delete, update, etc. vertices and edges, ensuring of integrity)
- to query graph; Gremlin is very efficient by querying the graph model
- exploring, analysis graphs
- exploring the semantic Web/Web of data; Gremlin can be used with RDF graphs and allows working with the semantic web in real-time
- gremlin is extensible with new methods and steps defined in Gremlin or in Java; Gremlin can take advantage of Java API
- it is a Turing complete language – it provides memory and computing constructs to support arbitrary path recognition

Simple Query Example of Gremlin to traverse the Graph: Gets all names and paths from vertex with ID = 1 (in Gremlin we have to choose arbitrary *root* vertex. The root vertex is the vertex from which searching starts. We can choose more than one vertex. Letter *g* is reference to the graph instance.

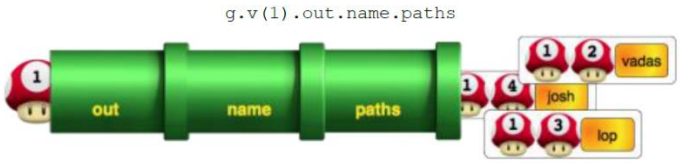


Figure 4: Gremlin Example

The Power of Graph Databases

A Graph Database has an "index-free adjacency" [4] mechanism to cross the graph without any index lookup. This means that once you have a record, to access related records you don't have to lookup relations in an index. – Like in traditional RDBMS – since relations are self-contained in the records themselves. Having self-contained relations means that **to move from a record to another one will always have a constant cost**, no matter how big the graph is: on the other end, RDBMS, once they start having a big amount of records, tend to highly worsen in terms of performances, since their indexes – and the lookups associated to them – grow logarithmically; in graph DBs, the cost is constant instead.

[4] IBM System G: *Graph Database Overview* [online], last visited 30.12.2015
http://systemg.research.ibm.com/database.html

OrientDB

OrientDB[5] is a tool capable of defining, persisting, retrieving and traversing information. We want to start there, rather than saying it is a type *ABC* database. Because OrientDB can be used in multiple ways. It can play a document database (making it a competitor to MongoDB, CouchDB, etc.), it can be a graph database (making it a competitor to Neo4J, Titan, etc.) and it can be an Object-Oriented Database. And it can play all those roles at the same time. It combines all the features of four model and make one complete core model. OrientDB continuously working to provide one solution for all types of NoSQL Database Models. OrientDB has three type of interfaces to work with: Console, OrientDB Studio and Gremlin console.

Features [6]

The Standard Edition is shipped with a rich set of out of the box features; all of them are immediately available after the server installation.

1- Apache2.0 license
2- ACID Transaction
3- Free of cost
4- Gremlin Language for graph computing
5- SQL Language Syntax for graph computing
6- RESTful API
7- Fast
8- Multi Master Replication
9- Sharding
10- Official release APIs for JAVA, .Net, PHP and many others
11- Developed in JAVA hence can be run in any OS.

The Document Model:

Documents is stored in this type of model. It does not forced to have schema. It also helps to created relationship between documents. Documents is stored in the form of Classes and Clusters and their relationship is represented as Link.

The table below illustrates the comparison between the relational model, the document model, and the OrientDB document model:

Relational Model	Document Model	OrientDB Document Model
Table	Collection	Class or Cluster
Row	Document	Document
Column	Key/value pair	Document field
Relationship	not available	Link

Table 1: Comparison between Document Model and Relational Model

The Object Oriented Model:

With OrientDB we are able to define a hierarchy between tables (they are called “classes”) and thus being able to take advantage of inheritance. Suppose we have collection of Animals and want to

[5] OrientDB [online], last visited 30.12.2015 http://orientdb.com/
[6] OrientDB Key advantages, [online] last visited 30.12.2015 http://orientdb.com/why-orientdb/

iterate through and output their calls. Different animals have different characteristics what if we present together in same table? Representing data in this way is a bad smell (called **NULLfull antipattern**, as it leads to records full of NULL attributes), but having different tables is not always a viable solution.

ID	TYPE	NAME	FERAL	WHISKERS	...
1	CAT	FURRY	0	LONG	...
2	DOG	JACOB	1	NULL	...
3	COBRA	NULL	NULL	NULL	...

Single Table Structure

CATS

ID	TYPE	NAME	FERAL	WHISKERS	...
1	CAT	FURRY	0	LONG	...

DOGS

ID	TYPE	NAME	FERAL	...
1	DOG	JACOB	0	...

SNAKES

ID	TYPE	VENOMOUS	...
1	COBRA	1	...

Multiple Table Structure

What if we have to query all animal which start with 'F' in multiple table scenario you have query in all table means N queries and at the end data have to combine results using UNION. In OrientDB you can simply create 'N' classes (Cats, Dogs, Snakes...) which extend parent class 'Animal'. Now the query seems little easy.

Select name from Animal where name like 'F'

The table below illustrates the comparison between the relational model, the Object model, and the OrientDB Object model

Relational Model	Object Model	OrientDB Object Model
Table	Class	Class or Cluster
Row	Object	Document or Vertex
Column	Object property	Document field or Vertex/Edge property
Relationship	Pointer	Link

Table 2: Object Oriented Model Representation in OrientDB

The Key/Value Model:

This is the simplest model of the three. Everything in the database can be reached by a key, where the values can be simple and complex types.

Relational Model	Key/Value Model	OrientDB Key/Value Model
Table	Bucket	Class or Cluster
Row	Key/Value pair	Document
Column	not available	Document field or Vertex/Edge property
Relationship	not available	Link

Table 3: Key/Value Model Representation in OrientDB

The Graph Model

And now comes Graph data model. As we discussed above graph database is form of Vertex and Edges. Representation of a Vertex is composed of a unique identifier, collection of properties, set of Incoming Edges (inE) and set of outgoing edges (outE) similarly an Edge is also composed of a unique identifier, an outgoing vertex (outV), a label, and incoming vertex (inV) and collection of properties which represents relationship between vertices shown in *(Figure 5)*. Each vertex or Edge can be any type of class which describes the structure and properties of vertex or edge and the class should inherit from base class V for Vertex and E for Edge respectively shown in *(Table 4)*.

A cluster is a place where a group of records are stored. OrientDB arranges create bunch of record per class. All the records of class are stored in one class. In OrientDB each record represent by its own unique identifier #<cluster-id>:<cluster-position> e.g. #12:0.

OrientDB support Bidirectional edges in OrientDB property graph model. OrientDB also supports Graph Language Gremlin as discussed above and can be usable from Gremlin console, OrientDB Studio or directly from Java API. Gremlin provides methods for working with graphs from Java API.

Relational Model	Graph Model	OrientDB Graph Model
Table	Vertex and Edge Class	Class that extends "V" (for Vertex) and "E" (for Edges)
Row	Vertex	Vertex
Column	Vertex and Edge property	Vertex and Edge property
Relationship	Edge	Edge

Table 4: Graph Model Representation in OrientDB

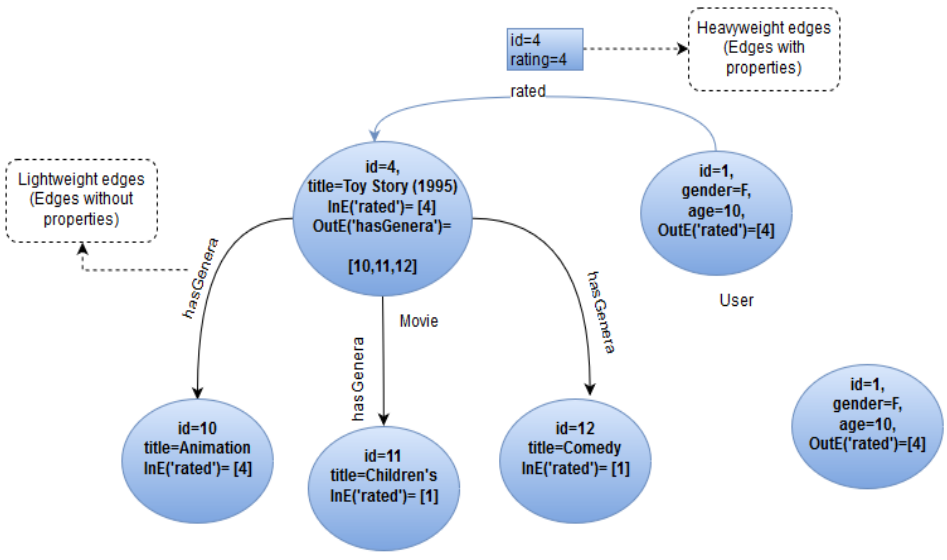
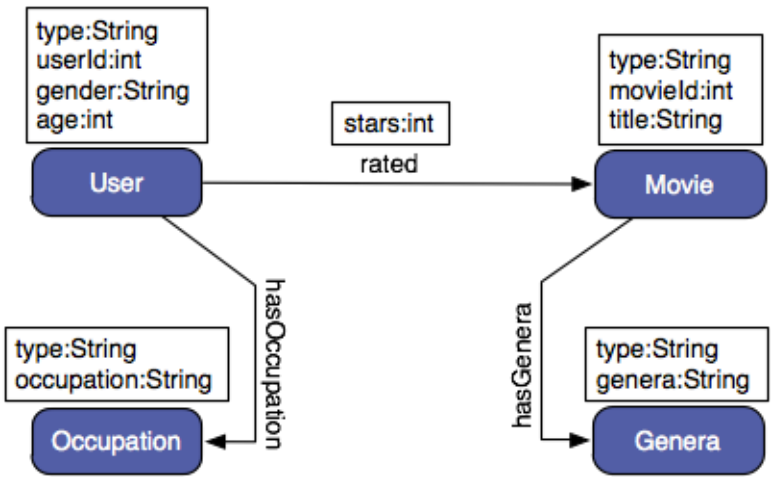


Figure 5: Representation of Vertex with edges (relationship) of MovieLens

Building Recommendation Engine in OrientDB

To work with OrientDB we choose MovieLens Dataset from GroupLens Research[7]. We used 1M[8] MovieLens dataset, contain 1,000,209 anonymous ratings of approximately 3,900 movies made by 6,040 users to build Movies Recommendation engine in both OrientDB 2.1.8 Community edition[9] and SQL Server 2012. Relational Diagram of this dataset is shown as below:



A recommender engine helps a user find novel and interesting items within a pool of resources. There are numerous types of recommendation algorithms and a graph can serve as a general-purpose substrate for evaluating such algorithms. We will demonstrate how to build a graph-based movie recommender engine using the MovieLens dataset. The following steps are used to build recommendation engine.
To load data in OrientDB we tries to explore more than one feature of OrientDB as show below.

Step 1 (a): Import Data using ETL

OrientDB comes with feature of ETL [10](Extract-Transform-Load) to load any type of files in OrientDB. It is based on configuration file of type .JSON[11]. Configuration File allows one extractor from source, multiple transformation and one destination.

Step 1(b): Import Data Using JAVA API

OrientDB developed in JAVA and comes with its more powerful native API[12]. You can download movies recommendation project here in JAVA developed by Davor Lozić[13] and explain step by step.

---

[7] GroupLens Department CSE at the University of Minnesota http://grouplens.org/datasets/movielens/
[8] MovieLens dataset 1M http://grouplens.org/datasets/movielens/1m/
[9] OrientDB download http://orientdb.com/download/
[10] OrientDB Manual Chapter 4. ETL http://orientdb.com/docs/2.0/orientdb-etl.wiki/Introduction.html
[11] JavaScript Object Notation https://en.wikipedia.org/wiki/JSON
[12] OrientDB JAVA Tutorial http://orientdb.com/docs/2.1/Tutorial-Java.html
[13] MovieLens JAVA http://warriorkitty.com/site/importing-movielens-into-orientdb-graph-database/

Step $1_{(c)}$: Import Data Using .NET API

During the working on OrientDB we found many examples implemented in Java that's why we planned to implement it in .Net to learn more about the official release of OrientDB .Net Driver[14].
The development is really interesting for .Net developers by using LINQ expression easily perform CRUD operations on database. Full source code can be downloaded from here.

Limitations .Net API:

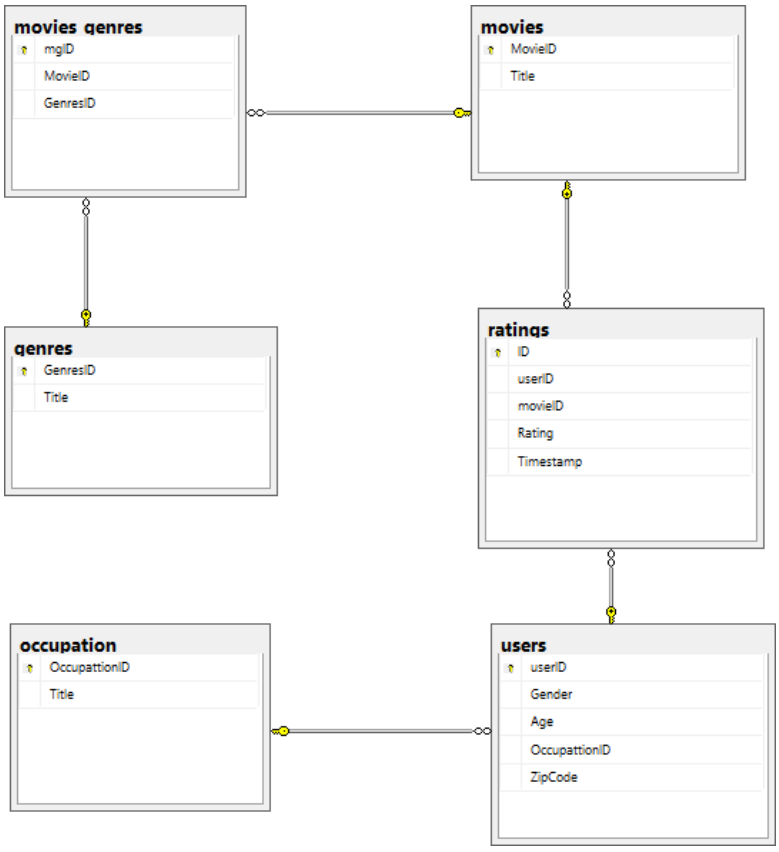
We test the .Net script on Core i7-6500U CPU 2.5GHZ, 8GB RAM Window 10. Performance issue was found while creating the 1M relations (edges) from users to movies which takes around ~20mins to load only 1M records. Luckily, OrientDB now have feature of MassiveInsert[15] available in Java API and OrientDB console.

Step 2: Import Data in SQL SERVER 2012

To compare the OrientDB queries with RDBMS. We created script of BULK INSERT, which can be downloaded from here.

Relational Logical Model of MovieLens

Following figure represents MovieLens RDBMS model in SQL Server 2012.



[14] .NET driver for OrientDB; Official Driver https://github.com/orientechnologies/OrientDB-NET.binary
[15] OrientDB Massive Insert Intent [online] http://orientdb.com/docs/2.1/Console-Command-Declare-Intent.html

Step 3: Query Graph Data Model:

To run the query you have to first start the OrientDB Server and launch OrientDB Studio in web browser using the following address http://localhost:2480/. Select Database MovieRaings and Open the Browser tabs to write queries.

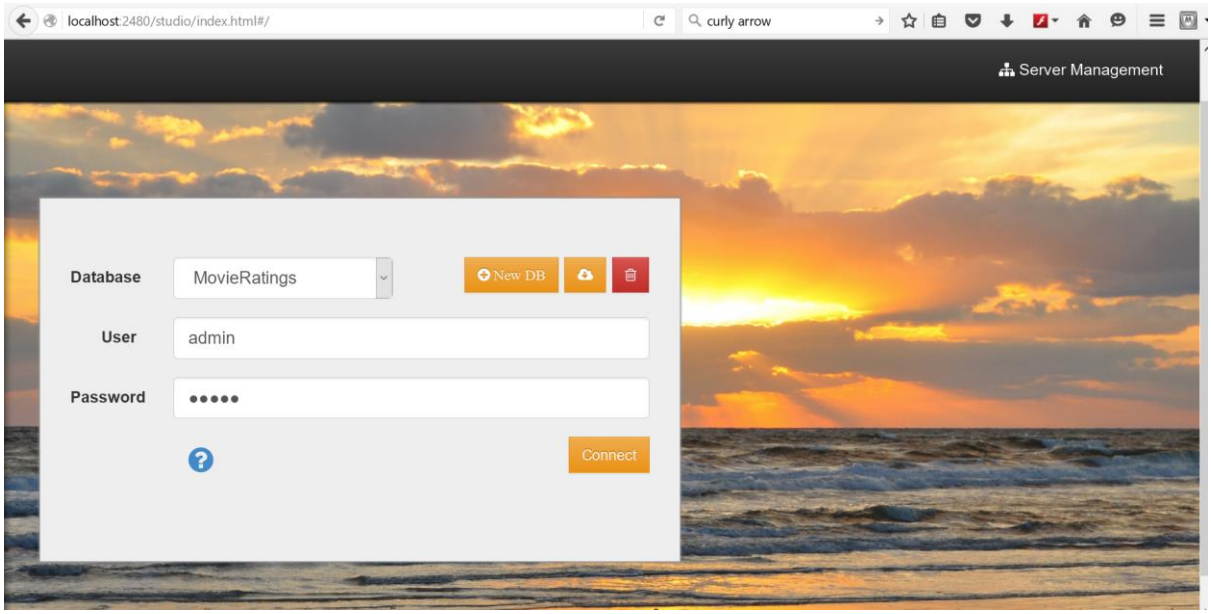


Figure 6: OrientDB studio

SQL

OrientDB's query language is built on SQL and is augmented with a few extensions to manipulate trees and graphs. Considering most developers are familiar with SQL, working with OrientDB is just easier and enhanced it with operators for graph manipulations. The Graph manipulation syntax derived from JDO/JPA standard[16] .

OrientDB SQL supports these constructions:
WHERE conditions, SELECT projections, TRAVERSE to cross records by relationships, INSERT, UPDATE, DELETE, Create Vertex/Edge to work with graphs, GRANT, REVOKE, Create class/property, Alter class/property, Create index, Rebuild index, Create link, Alter cluster and next

Gremlin console also available in OrientDB to query using Gremlin.

Building Queries

Let's start with basic syntax to write SQL query in OrientDB. We found in most of the scenario OrientDB SQL and SQL server query syntax almost same.

[16] JAVA Data Objects https://db.apache.org/jdo/jdo_v_jpa.html

OrientDB | Microsoft SQL Server 2012

1- **Select user with id=1?**

```
Select * from users where id=1
```

@class	id	gender	age	occupationId	zipCode
Users	1	F	1	10	48067

```
Select * from users where userID=1
```

Results | Messages

	userID	Gender	Age	OccupattionID	ZipCode
1	1	F	1	10	48067

2- **How many movies (vertex) available in MovieLens Dataset?**

```
Select count(*) from Movies
```

PROPERTIES
count
3883

```
Select count(*) from Movies
```

Results | Messages

	(No column name)
1	3883

3- **No More Joins in OrientDB**
Movie Toy Story belongs to which Genera's?

```
Select expand (outE('hasGenera').in.description)
from movies
where id=1
```

value
Animation
Children's
Comedy

```
Select Title
from movies_genres mg  join genres g
on mg.GenresID= g.GenresID
where MovieID=1
```

Results | Messages

	Title
1	Animation
2	Children's
3	Comedy

4- **Power of Group by**
What is the distribution of occupations amongst the user population?

```
Select description, count(*)
from
    ( Select expand( out('hasOccupation'))
      from Users)
Group by description
Order by description
```

```
Select Title, count(userID) as C from users u join
occupation occ
on u.OccupattionID=occ.OccupattionID
group by Title
order by Title
```

description	count
academic/educator	528
artist	267
clerical/admin	173
college/grad student	759
customer service	112
doctor/health care	236
executive/managerial	679
farmer	17
homemaker	92
K-12 student	195

	Title	C
1	academic/educator	528
2	artist	267
3	clerical/admin	173
4	college/grad student	759
5	customer service	112
6	doctor/health care	236
7	executive/managerial	679
8	farmer	17
9	homemaker	92
10	K-12 student	195
11	lawyer	129
12	other or not specified	711

5- **Which user give maximum rating to movies?**

```
select id, outE('rated').size() as C
from users
order by C desc
limit 1
```

id	C
4169	2314

Description: user id 4196 gave rating to 2314 movies.

```
Select top 1  userID, COUNT(movieID) [count]
from ratings
Group by userID
order by [count] desc
```

userID	count
4169	2314

6- **Users gave 3 stars to Toy Story (1995) and same users gave 3 stars to which other movies?**

```
Select expand( inE('rated')[rating = 3]
               .outV().OutE('rated')[rating=3]
               .inV().title)
from #13:0
```

title
Toy Story (1995)
Day the Earth Stood Still, The (1951)
Raise the Red Lantern (1991)
Varsity Blues (1999)
Ice Storm, The (1997)
Chasing Amy (1997)
Buena Vista Social Club (1999)
Hud (1963)
Anatomy of a Murder (1959)
Two Women (La Ciociara) (1961)

10 25 50 100 1000 5000

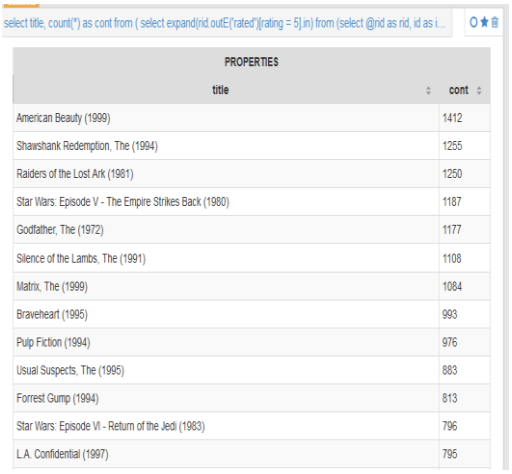
Description: Notice that this list has many duplicates. This is due to the fact that users who like Toy Story also like many of the same other movies.

```
Select Title from movies m join (
   Select r.movieID from ratings r join
      (Select userID from ratings where Rating=3
       and movieID=1) TSRating
   on r.userID=TSRating.userID
   where r.Rating=3) r1
on m.MovieID=r1.movieID
```

	Title
1	Aladdin (1992)
2	Twister (1996)
3	Strictly Ballroom (1992)
4	George of the Jungle (1997)
5	Shawshank Redemption, The (1994)
6	American President, The (1995)
7	Negotiator, The (1998)
8	Home Alone (1990)
9	Last of the Mohicans, The (1992)

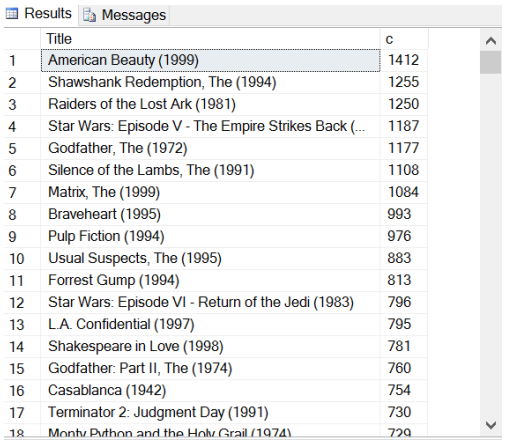
7- **Among the users similar to the user id =1 (#16:0), which film has received more 5 stars and is still not present in the films rated by 16:0**

```
Select title, count(*) as cont
from ( select expand(rid.outE('rated')[rating = 5].in)
   from ( select @rid as rid, id as id, count(*) as cont
     from ( select expand(outE('rated')[rating=5]
          .in.inE('rated')[rating=5].out) from #16:0)
     where @rid <> #16:0 group by rid, id
     order  by id))
 where title not in (select out('rated').title from #16:0)
 group by title
 order by cont desc
```



PROPERTIES	
title	cont
American Beauty (1999)	1412
Shawshank Redemption, The (1994)	1255
Raiders of the Lost Ark (1981)	1250
Star Wars: Episode V - The Empire Strikes Back (1980)	1187
Godfather, The (1972)	1177
Silence of the Lambs, The (1991)	1108
Matrix, The (1999)	1084
Braveheart (1995)	993
Pulp Fiction (1994)	976
Usual Suspects, The (1995)	883
Forrest Gump (1994)	813
Star Wars: Episode VI - Return of the Jedi (1983)	796
L.A. Confidential (1997)	795

```
Select m.Title, COUNT(*) c from movies m join (
    Select  r4.movieID, r4.userID from ratings r4 join (
       Select  userID, count(*) cont from  (
          Select r1.ID, r1.userID, r1.movieID, r1.Rating
          from ratings r1 join
              ( Select userID,movieID from ratings
               where userID=1 and Rating=5)r2
          on r1.movieID=r2.movieID
          where r1.Rating=5)r3
       where r3.userID !=1
       group by r3.userID) r5
  on r4.userID=r5.userID
  where Rating=5) r6
on m.MovieID=r6.movieID
where m.MovieID NOT IN
          ( SELECT movieID from ratings where userID=1 )
group by Title
order by c desc
```



Results | Messages

	Title	c
1	American Beauty (1999)	1412
2	Shawshank Redemption, The (1994)	1255
3	Raiders of the Lost Ark (1981)	1250
4	Star Wars: Episode V - The Empire Strikes Back (...	1187
5	Godfather, The (1972)	1177
6	Silence of the Lambs, The (1991)	1108
7	Matrix, The (1999)	1084
8	Braveheart (1995)	993
9	Pulp Fiction (1994)	976
10	Usual Suspects, The (1995)	883
11	Forrest Gump (1994)	813
12	Star Wars: Episode VI - Return of the Jedi (1983)	796
13	L.A. Confidential (1997)	795
14	Shakespeare in Love (1998)	781
15	Godfather: Part II, The (1974)	760
16	Casablanca (1942)	754
17	Terminator 2: Judgment Day (1991)	730
18	Monty Python and the Holy Grail (1974)	729

8- **Recommendation by Genre: Find the top 5 genre interest of user (#16:0) and recommend more movies to like of that genre which is not yet rated**

```
Select description, count(*)
from (select expand(in.out('hasGenera'))
        from ( select expand(outE()) from #16:0)
        where rating > 3)
group by description order by count desc
limit 5
```

```
Select top 5 g.Title, count(*) c  from genres g join
     ( Select mg.MovieID, GenresID
       from movies_genres mg join
            ( select movieID from ratings where userID=1
             and Rating>3)m
        on mg.MovieID=m.movieID) mg2
 on g.GenresID=mg2.GenresID
 group by Title
 order by c desc
```

description	count
Drama	21
Children's	18
Animation	14
Musical	12
Comedy	11

Results | Messages

	Title	c
1	Drama	21
2	Children's	18
3	Animation	14
4	Musical	12
5	Comedy	11

9- **Suggest top 5 movies rated with 5 stars to the user's most favorite genres**

```
Select title, count(*)
 from (
     select expand(rid.in().inE('rated')[rating = 5].in)
     from ( select @rid, description, count(*)
            from ( select expand(in.out('hasGenera'))
                   from ( select expand(outE())
                          from #16:0)
                   where rating > 3)
            group by @rid, description
            order by count desc limit 5))
 where title not in
     (select out('rated').title from #16:0)
group by title
order by count desc
Limit 5
```

title	count
American Beauty (1999)	3926
Babe (1995)	1689
This Is Spinal Tap (1984)	1545
Star Wars: Episode V - The Empire Strikes Back (1980)	1483
Godfather, The (1972)	1475

```
Select Title, c as RatedUserCount
from movies m1 join (
   Select top 5  r1.movieID, count(*) c
    from ratings r1  join
        ( Select  MovieID  from movies_genres
          where GenresID IN
             ( Select GenresID from
                ( Select top 5 g.GenresID, count(*) c
                 from genres g join
                      (Select mg.MovieID, GenresID  from
                       movies_genres mg join  (
                          select movieID from ratings
                           where userID=1 and Rating>3)m
                       on mg.MovieID=m.movieID) mg2
                  on g.GenresID=mg2.GenresID
                  group by g.GenresID
                  order by c desc) MostLikes))r2
       on r1.movieID=r2.MovieID
       where r1.Rating=5 and  r1.movieID Not in
           (select movieID from ratings where userID=1)
      group by r1.movieID
     order by c desc) m2
on m1.MovieID=m2.movieID
order by c desc
```

Results | Messages

	Title	RatedUserCount
1	American Beauty (1999)	3926
2	Babe (1995)	1689
3	This Is Spinal Tap (1984)	1545
4	Star Wars: Episode V - The Empire Strikes Back (...	1483
5	Godfather, The (1972)	1475

Code Sample

```
using Orient.Client;

namespace Import_DataSet {

    class Program {

        static void Main(string[] args){

           Try {
                  // create Database connection
                OrientDBDriver.CreateDatabase();
                OrientDBDriver.CreatePool();

                ODatabase oDB = new ODatabase(OrientDBDriver.DatabaseAlias);

                // Create Vertex and Edges and Load movielens dataset file

                oDB.Create.Class<Occupations>()
                          .Extends<OVertex>()
                          .CreateProperties<Occupations>()
                          .Run();

                List<Occupations> _lisOccupations = LoadOccupations();

                foreach (Occupations occ in _lisOccupations) {
                    oDB.Create.Vertex<Occupations>(occ).Run();
                }

                List<Users> _listUsers = LoadUsers(); // loading user.dat

                oDB.Create.Class<Users>().Extends<OVertex>()
                                         .CreateProperties().Run();

                oDB.Create.Class("hasOccupation").Extends<OEdge>().Run();

                foreach (Users user in _listUsers) {
                    ODocument odoc = oDB.Create.Vertex("Users")
                                    .Set("userID", user.userID)
                                    .Set("Gender", user.gender)
                                    .Set("Age", user.age)
                                    .Set("ZipCode", user.ZipCode)
                                    .Run();
                    string generateSQL = new OSqlSelect().Select()
                                                         .From("Occupations")
                                                         .Where("OccupationID")
                                                         .Equals(user.Occupation
                                                         ID)
                                                         .Limit(1).ToString();
                    List<ODocument> result = oDB.Query(generateSQL);
                    if (result.Count > 0) {
                        OEdge edge = oDB.Create.Edge("hasOccupation")
                                    .From(odoc.ORID)
                                    .To(result[0].ORID)
                                    .Run();
                    }
                }
```

Performance Tuning

Many aspects can be considered to get better performance. Let's start with general introduction to some basic concepts about how OrientDB uses the memory and how it manages the I/O.

Caching:

OrientDB uses two caches L1 (thread level) and L2 at JVM Level. If you are in a multi-threaded scenario and tries to perform many writes; you may disable the L1 cache. Or, again if you are in a multi-JVM scenario, you may consider disabling the level 2 cache also.[17]

Connection:

Local connection is much faster than remote. Use "plocal" based on the storage used on database creation.

Query and DB Structure Tips:[18]

- Avoid putting properties on edges.
- It's much lighter to set properties in block than one by one
- It's even faster if you set properties directly on creation of vertices and edges
- For bulk operation in JAVA API or Console use MassiveInsert
- Use indexes to lookup vertices by an ID
- Disable validation

[17] Claudio Tesoriero (2013), *Getting Started with OrientDB* Chapter 4: Performance Tuning
[18] OrientDB Documentation Performance Tuning [online], last visited 30.12.2015 http://orientdb.com/docs/2.1/Performance-Tuning.html

Installing OrientDB:

OrientDB is available in community and enterprise edition. Community addition is released as open source under Apache 2.0 licence.

OrientDB in written in Java, so you can run it on any platform, it requires version 1.6 or higher. It comes with server, console and Gremlin and also OrientDB Studio. 3-Steps installation required for all operating systems. [19]

For Windows

1- Download .zip file for windows and extract.
2- Go to bin directory and run server.bat.
3- Now you can start the Console.bat to run queries in console or go to http://localhost:2480/ for OrientDB studio.

For Mac

1- Download .zip file for windows and extract.
2- Go to bin directory and run server.sh.
3- Now you can start the Console.bat to run queries in console or go to http://localhost:2480/ for OrientDB studio.

For Linux and any other *NIX system

For Linux and Unix distributions that rely on init, copy the edited system daemon file to the ***/etc/init.d/***. To make the console accessible, copy the console script file to the system binary directory ***/usr/bin***

```
$ cp $ORIENTDB_HOME/bin/orientdb.sh /etc/init.d/orientdb

$ cp $ORIENTDB_HOME/bin/console.sh /usr/bin/orientdb
```

Doing this makes the script accessible to the service command. You can now start the database server using the following command:

```
$ service orientdb start
```

Once the database starts, it is accessible through the console script.

[19] OrientDB Manual Installation, [online] last visited 30.12.2015 http://orientdb.com/docs/2.1/Tutorial-Installation.html

Conclusion

In conclusion we would like to share our feedback of OrientDB not only Graph Database but also Multimodal NoSQL Database. If we individually take each of this feature, we won't get excited, as most of the products in the DB market implement a few of them, but being able to meld down all of them together OrientDB is simply something that no developer has ever seen before; in his brief history, it has gained so much attention that almost everyone in the NoSQL ecosystem is looking at this new competitor with a curious eye.

OrientDB won't be Swiss-army knife, is not going to be the *one-size-fits-all* tool you always needed and never found before: it is a new way to think about data in our times, a way that has its own boundaries and scopes.

On top of this, OrientDB is not only a NoSQL database: it's a mixture of RDBMS, NoSQL databases and eventually a graph DB; what makes this product so interesting is that it melds together 3 worlds as it never happened before. OrientDB focuses on performances and it has been built to extremely optimize data retrieval operations.

While working with OrientDB, we found its worth to know about new technology definitely in future projects OrientDB as a Graph database will be our first preference for development of NoSQL database application.

It was really very exciting and tremendous experience in exploring OreintDB technologies and the NoSQL technologies especially with focus on Graph Databases.