2012 Third International Conference on Emerging Intelligent Data and Web Technologies

MongoDB vs Oracle - database comparison

Alexandru Boicea, Florin Radulescu, Laura Ioana Agapin
Faculty of Automatic Control and Computer Science
Politehnica University of Bucharest
Bucharest, Romania
alexandru.boicea@cs.pub.ro, florin.radulescu@cs.pub.ro, lauraioana.agapin@gmail.com

***Abstract* — This paper presents differences between an SQL database management system - Oracle Database and NoSQL document oriented database management system - MongoDB. Comparison criteria includes theoretical differences, features, restrictions, integrity, distribution, system requirements, architecture, query and insertion times.**

Keywords: instruction, database, nosql, document oriented, function

I. INTRODUCTION

The following work presents the main differences between MongoDB and Oracle database management systems as seen from a typical user point of view, starting from the system requirements, the operating systems you can install them on, syntax differences, drivers, to differences of performance regarding query time, insert time and update time on a identical test database.

We choose MongoDB as a NoSQL database management system because it is relatively new on the database market and it is used in many important projects and products, such as: MTV Networks, Craiglist, Disney Interactive Media Group, Sourceforge, Wordnik, Yabblr, SecondMarket, The Guardian, Forbes, The New York Times, bit.ly, GitHub, FourSquare, Disqus, PiCloud etc. Every one from this list uses MongoDB because they found something that fits very good with their projects. For example, Craiglist uses MongoDB for his archiving bilions of records, MTV uses MongoDB as main repository for MTV Networks, on SourceForge, MongoDB is used for back-end storage, The Guardian uses MongoDB because they discovered that they can store documents very simple using this type of database management system, The New York Times uses MongoDB in a form-building application for photo submissions, bit.ly uses MongoDB to store user history.

Oracle Database is an object-relational database management system produced and marketed by Oracle Corporation. It is one of the oldest SQL databases. The development process for Oracle Database started in 1977. It is one of the most common used and reliable relational database management systems.

II. MAIN DIFFERENCES

The main difference is that NoSQL is a category of database engines that do not support SQL in order to achieve performance or reliability features that are incompatible with their flexibility of SQL. These engines usually provide a query language that provides a subset of what SQL can do, plus some additional features. JOIN, TRANSACTION, LIMIT and non-indexed WHERE usually are not supported by NoSQL engines.

Some NoSQL engines have automatic import-from-SQL features, bust using NoSQL imposes some architecture constraints that are depending of what you are doing with the data. It is rare for an entire existing SQL-driven application to be moved in its integrality over to NoSQL.

NoSQL is a class of database management system different from the traditional relational databases in that data is not stored using fixed table schemas. Mainly its purpose is to serve as database system for huge web-scale applications where they outperform traditional relational databases.

MongoDB is a NoSQL database management system released in 2009. It stores data as JSON-like documents with dynamic schemas (the format is called BSON). MongoDB has his focus oriented to four things: flexibility, power, speed and ease of use. It supports replicated servers and indexing and it offers drivers for multiple programming languages.

Another difference between MongoDB and Oracle Database is the database model: MongoDB is a

978-0-7695-4734-3/12 $26.00 © 2012 IEEE
DOI 10.1109/EIDWT.2012.32

IEEE computer society

document-oriented NoSQL schema-less database model and Oracle is a relational database model. The query language used by the Oracle Database is SQL in comparison with the one used by MongoDB which consists in API calls, JavaScript and REST.

Both MongoDB and Oracle Database use conditional entry updates, composite keys, unicode characters and full text search. MongoDB has aggregation functions. A built-in map-reduce function can be used to aggregate large amounts of data.

MongoDB accepts larger data. The maximum value size in MongoDB is 16 MB in comparison with Oracle Database which maximum value size is 4 KB.

The integrity model used by Oracle Database is ACID, while MongoDB uses BASE. MongoDB offers consistency, durability and conditional atomicity. Oracle Database offers integrity features that MongoDB doesn't offer like: isolation, transactions, referential integrity and revision control.

In manners of distribution both MongoDB and Oracle Database are horizontal scalable and have support for data replication. While MongoDB offers sharing support, Oracle Database doesn't.

Both MongoDB and Oracle Database are cross platform database management systems. Oracle Database was written in C++, C and Java, while MongoDB was written in C++. Both engines are multi user and active databases. MongoDB is a free product, while Oracle Database is a licensed product. None of the databases have compression support.

III. SYNTAX DIFFERENCES

There are big differences of syntax between the two database management systems.

Oracle Database uses common SQL language. It uses data manipulation statements as SELECT, UPDATE, DELETE.

MongoDB uses functions for the operations of adding new records, updating and deleting existing records.

Oracle Database offers procedures to manipulate data returned of the select. The language used for advanced queries is PL/SQL, as in procedural language, structured query language.

For this kind of advanced queries, MongoDB uses callback functions. For the mongo shell, the language used for declare functions is JavaScript. These functions are used to manipulate data returned by the basic queries. One can iterate through the results of the queries and apply different functions.

For better understanding the differences between the two database syntax, we will give some examples.

First of all we will talk about creating an object in which we can insert data. This object is called table in Oracle Database and collection in MongoDB.

In Oracle Database the creation of a table is made using the statement CREATE:

```
CREATE TABLE users(
   user_id INT NOT NULL,
   first_name VARCHAR2(50),
   last_name VARCHAR2(50)
);
```

In MongoDB the creation of the collection is made on the first insert. Because of the flexibility of the MongoDB collection, the structure doesn't have to be fixed.

Droping a table with all the constraints or indexes can be very easy.

In Oracle Databases is a function that will drop a table if there are not other tables related to this one:

```
DROP TABLE users;
```

In MongoDB there are no dependencies between the tables so you can easily drop any table and the indexes assigned to it:

```
db.users.drop();
```

Regarding data insertion, Oracle Databases have multiple types of constraints on tables. When you are trying to insert a new record you have to be careful not to violate any of the constraints. On the other hand, MongoDB is more flexible with the data. You don't have any constraints regarding the data from the collections. This is the major difference between a SQL Database and a NoSQL one.

The SQL data is stored into tables with a fixed structure. One can define relations between the tables from the same database. The relations between the tables can be foreign key relations. You can set PRIMARY KEY constraint on the tables. One table can have a primary key composed of one or more columns from the table. There can be other constraints declared on the columns of a table, like UNIQUE, FOREIGN KEY or NOT NULL.

In NoSQL databases data is stored using collections. MongoDB collections have no constraint regarding the data stored in them. The collections don't have a fixed structure. The fields can have different data types.

In Oracle Databases a new record is saved into the database using the INSERT statement. For example, if you want to insert a new record in the “users” table, the statement looks like this:

```
INSERT INTO users( id_user, first_name, last_name )
VALUES( 1, ‘Alin’, ‘DUMITRU’ );
```

In MongoDB the insertion of a new record is made using functions and BSON objects. The functions used for inserting new data are ‘insert’ and ‘save’. When inserting a new object into a MongoDB collection, a new field is automatic added to your new object. That field is called ‘_id’ and is unique and used like a index. For example, if you want to add the same user into the ‘user’ collection from the MongoDB database, you have to write in the mongo shell something like this:

```
db.users.insert( { ‘user_id’: 1, ‘first_name’: ‘Alin’, ‘last_name’: ‘DUMITRU’ } );
```

or:

```
db.users.save( { ‘user_id’: 1, ‘first_name’: ‘Alin’, ‘last_name’: ‘DUMITRU’ } );
```

The basic data retrieval in the Oracle Database is made using the simple SELECT statement. One will specify the fields that he is interested in and the tables from which he wants to retrieve data. Here is an example of a simple SELECT statement from a single table:

```
SELECT user_id, first_name, last_name
FROM users;
```

The output for this statement is a table containing all the users from the users table. If you want to refine the search, you can include a WHERE statement.

For example, if you want to retrieve only the users which last name is Stevens, you should include a WHERE clause like the following:

```
WHERE last_name LIKE 'Stevens';
```

You can order the results using the ORDER BY clause. The results can be ordered ascending or descending.

In MongoDB, you can retrieve results using the “find” function. In MongoDB, there aren't any tables. The data is stored using collections of data. This collections don't have a fixed structure. The items stored in this collections can have different fields.

For example, supposing we have a collection of users named “users”. If you want to retrieve all the items from this collection you can use something like this:

```
db.users.find();
```

If you want to retrieve only some of the items in the collection, for example only the users who has the first name “Stevens” you use the “find” function with an extra object:

```
db.users.find( { first_name: “Stevens” } );
```

If you don't want your query to display all the fields, you can add an object to specify which fields to be displayed. For the “_id” field, which MongoDB automatically adds to new objects inserted into the collections, you have to specify not to display it; for the other fields you have to specify to display them.

For example, if you want to display only the “last_name” and “first_name” and not the “user_id” and the “_id”:

```
db.users.find( {}, { “_id”: 0, “last_name”: 1, “first_name”: 1 } );
```

In MongoDB you can sort the data. You can use the sort function where you specify the field you want to sort the data by and the sorting order: ascending (1) or descending (-1):

```
db.users.find().sort( { “last_name”: 1 } );
```

The removing of data is usually made using some criteria or not.

In Oracle Database, the data removing from a table is made using the DELETE statement:

```
DELETE FROM users;
```

If you want to delete, for example, only the rows which have the ‘user_id’ greater then 10, the statement should look like this:

```
DELETE FROM users WHERE user_id > 10;
```

In MongoDB collections, the function used for deleting the content is ‘remove’:

```
db.users.remove();
```

If you want to remove only some entries, like the ones having the ‘user_id’ greater then 10, then you should write somenthing like this:

```
db.users.remove( { ‘user_id’: { ‘$gt’: 10 } } );
```

You can easily update an entry or part of an entry in either databases. In Oracle Databases, if you want to update an entire table you use the statement:

```
UPDATE users
SET last_name = ‘last_name’;
```

If you want to update only the users with the ‘user_id’ greater then 100, the statement looks like this:

```
UPDATE users
SET last_name = ‘last_name’
WHERE user_id > 100;
```

In MongoDB you can easily update one row or many rows. The command if you want to update the entire collection is:

```
db.users.update( {},
   {‘$set’: {‘last_name’: ‘last_name’}} );
```

The first object is used for selecting the entries which you want to update. In this case the query returns all the entries from the collection. If you want to update only the entries which have the ‘user_id’ greater then 100, the update function looks like this:

```
db.users.update( {‘user_id’: {‘$gt’: 100}},
   {‘$set’: {‘last_name’: ‘last_name’}} );
```

With the update function you can replace an object from the collection, you can pull an item from an array, you can push an item to an array, you can set a new field, you can update a set of fields, you can do a lot of things.

Although there are syntax differences, working with either databases is light and intuitive. Both databases have their advantages and disadvantages, but regarding the syntax, both of them are user friendly and easy to learn.

IV. TIME COMPARISON

For a best comparison between the two database engines, we run some tests and compute the time that took each engine to take some actions on the database.

The first test we did was regarding the INSERT operation. We generated objects to be inserted in the database. Each object has an ‘user_id’, a ‘last_name’ and a ‘first_name’. It was kept the same structure for both of the databases.

We took a set of 10 inserts and we computed how much it took to insert the rows in the database. After that we took 100 records, 1000, 10000, 100000, until we got to 1000000.

The insert for the Oracle Database engine looks like this:

```
INSERT INTO users
   ( user_id, last_name, first_name )
   VALUES( 1, 'last_name1', 'first_name1 );
```

The insert function used for MongoDB looks like this:

```
db.users.insert( {
   user_id: 1,
   last_name: 'last_name1',
   first_name: 'first_name1'
});
```

Table 1 show the number of millisecond for each bunch of inserts.

TABLE 1 –Inserting times (msec)

No. of records	Oracle Database	MongoDB
10	31	800
100	47	4
1000	1563	40
10000	8750	681
100000	83287	4350
1000000	882078	57871

As you can see from the table 1, MongoDB is more efficient when inserting a large amount of data. On the other hand, it took too little to long to insert only 10 records.

Oracle Database deals very nice with little amount of data, but when we talk of large amounts, greater then 10000 records, the time spent on this operation is bigger.

The Fig.1 shows how the time spent on the insert operation by Oracle Database is bigger.

After the insert operation, we would like to see how much does it take to update some records.

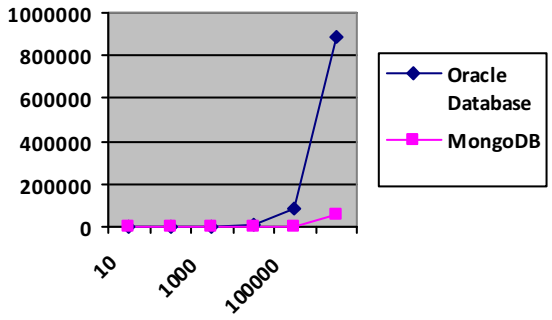

Figure 1. Inserting times(msec)

Just like the insert case, we tested both database engines to see how much does updates take. For this test we used the inserted records before.

For the first 10 updates, it was updated the records with the 'user_id' less then 10, for the 100 updates, we used the records with the 'user_id' between 11 an 110, for the 1000 updates the 'user_id' between 111 and 1110, for the 10000 the 'user_id' between 1111 and 11110, do the 100000 the 'user_id' between 11111 and 111110 and finally, for the 1000000 the 'user_id' between 111111 and 1111110. The field updated is 'last_name'.

The statement used to update the records on Oracle Database is:

```
UPDATE users
SET last_name = 'last_name100'
WHERE user_id > 10
AND user_id <= 110;
```

For MongoDB we used the update function like this:

```
db.users.update(
   { user_id: { '$gt': 10, '$lte': 110 } },
   { '$set': { last_name: 'last_name100' } } );
```

We computed the time that took for each database engine to do those requests and we put them in the table 2.

TABLE 2 – Updating times (msec)

No. of records	Oracle Database	MongoDB
10	453	1
100	47	1
1000	47	1
10000	94	1
100000	1343	2
1000000	27782	3

As you can see from the table 2, updating in MongoDB is nearly the same for 10, 100 or 1000000 records.

On Oracle Databases, the time took by the engine for updating the records is increasing with the number of records to be updated.

It is visible the difference between the Oracle Database and the MongoDB database systems.

We put those number into Fig.2 to see better the differences.

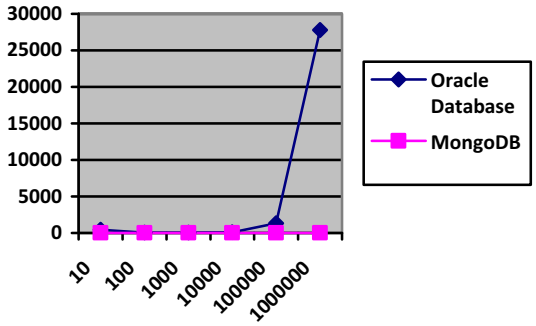

Figure 2. Updating times(msec)

Regarding the removal of the objects from the database, we used the same type of test. We removed 10 records, after that 100, 1000, 10000, 100000 and 1000000. We used for this test the collections created on the earlier tests.

Data removal on Oracle Database is made using the DELETE statement:

```
DELETE FROM users
WHERE user_id > 10
AND user_id <= 110;
```

On MongoDB databases, the removal of data is made using the 'remove' function:

```
db.users.remove(
   { user_id: { '$gt': 10, '$lte': 110 } } );
```

We computed the times for this requests and we put them into the table 3.

TABLE 3 – Deleting times (msec)

No. of records	Oracle Database	MongoDB
10	94	1
100	47	1
1000	62	1
10000	94	1
100000	1234	1
1000000	38079	1

The deleting times are shown in the Fig. 3 .

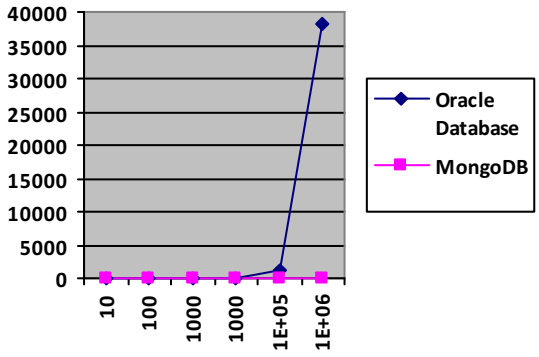

Figure 3. Deleting times(msec)

The differences between MongoDB and Oracle Database are huge. As you can see, the time spent by MongoDB deleting the records is almost constant, while on Oracle Database is increasing with the number of records to be deleted.

V. CONCLUSIONS

MongoDB provides flexibility during the development process. It has built in support for horizontal scalability. It is easy to deploy and copy databases from one server to another using export – import tools. You can store complex data into one field – you can store and object, an array or a reference into one field.

It maps easily some objects from different language problems into the database(like java script objects or python objects). It does not need any kind of conversion.

MongoDB is a more rapid database management system. If you want a simple database that will respond very fast, this is the choice you should make.

You can use the map-reduce function to aggregate results for reports. You can sum up the results, you can merge the fields, you can make whatever you want when you're doing map-reduce. It's fast and flexible.

Being open source, you can develop plugins to make it easy to work with. It is into a continuous developing program and you can find useful tips in the open source community.

On the other hand, if you need a more complex database, with relations between tables and a fix structure, you should stay with the classic Oracle Database. It's a reliable database and even thought it moves slower, it is the base for complex structured databases.

The main difference between the two databases is that Oracle Database has relations between the tables. Those relations can be one to one or one to many or many to many. With these relations you can join tables and make complex queries.

The main problem with the Oracle Database is the replication. You cannot copy the database so easily as in MongoDB. You have tools which do that but aren't so fast. It is a much slower database in comparison with MongoDB.

In conclusion, if you want to use a fast, flexible database, you can rely on MongoDB. If the rapidness of the database isn't you main concern, and if you need relations between the tables and the collections, you can rely on the classic solution, Oracle Database.

REFERENCES

[1] K. Banker, "MongoDB in action", 2011

[2] M. Rosenblum and P. Dorsey, "Oracle PL/SQL for dummies" , 2006

[3] http://www.mongodb.org/

[4] http://vschart.com/compare/oracle-database/vs/mongodb

[5] http://en.wikipedia.org/wiki/MongoDB

[4] http://en.wikipedia.org/wiki/Sql