**Module 2 – Schema Design and Data Modelling** - Case Study Solution – 2

Compare both the Database based on terminology, concepts , features, and query language

Many concepts in Oracle have close analogs in MongoDB. The table below outlines the common concepts across Oracle and MongoDB.

| Oracle | MongoDB |
| --- | --- |
| ACID Transactions | ACID Transactions |
| Table | Collection |
| Row | Document |
| Column | Field |
| Secondary Index | Secondary Index |
| JOINs | Embedded documents, $lookup & $graphLookup |
| GROUP\_BY | Aggregation Pipeline |

Feature Comparison

Like Oracle, MongoDB offers a rich set of features and functionality far beyond those offered by simple NoSQL data stores. MongoDB has a rich query language, highly-functional secondary indexes (including text search and geospatial), a powerful aggregation framework for data analysis, faceted search, graph processing and more. With MongoDB you can also make use of these features across more diverse data types than a relational database, and you can do it at scale.

|  | Oracle | MongoDB | NoSQL Data Store |
| --- | --- | --- | --- |
| ACID Transactions |  |  |  |
| Flexible, rich data model |  |  | Partial: schema flexibility but support for only simple data structures |
| Schema governance |  |  |  |
| Expressive joins, faceted search, graphs queries, powerful aggregations |  |  |  |
| Idiomatic, native language drivers |  |  |  |
| Horizontal scale-out with data locality controls |  |  | Partial: no controls over data locality |
| Analytics and BI ready |  |  |  |
| Enterprise grade security and mature management tools |  |  |  |
| Database as a service on all major clouds | Partial: AWS and Oracle cloud |  |  |

Query Language

Both Oracle and MongoDB have a rich query language. Below are a few examples of SQL statements and how they map to MongoDB. A more comprehensive list of statements can be found in the [MongoDB documentation](https://docs.mongodb.com/manual/reference/sql-comparison/?_ga=1.101830017.1883246389.1439224661).

|  |  |
| --- | --- |
| Oracle | MongoDB |
| INSERT INTO users (user\_id, age, status) VALUES ('bcd001', 45, 'A') | db.users.insert({ user\_id: 'bcd001', age: 45, status: 'A' }) |
| SELECT \* FROM users | db.users.find() |
| UPDATE users SET status = 'C' WHERE age > 25 | db.users.update( { age: { $gt: 25 } }, { $set: { status: 'C' } }, { multi: true } ) |
| db.start\_transaction() cursor.execute(orderInsert, orderData) cursor.execute(stockUpdate, stockData) db.commit() | s.start*transaction() orders.insert*one(order, session=s) stock.update*one(item, stockUpdate, session=s) s.commit*transaction() |

After comparison, I suggest to choose Plan-B, because MongoDB save Employees, Time and Resource requirements as well as it gives more performance as compared to Oracle.

#### **Configuration ?**

In order to work with MongoDB, first you need to install MongoDB on your computer. To do this, visit [the official download center](https://www.mongodb.com/download-center/community) and download the version for your specific OS. Here, I’ve used Windows.

After downloading MongoDB community server setup, you’ll go through a ‘next after next’ installation process. Once done, head over to the C drive in which you have installed MongoDB. Go to program files and select the MongoDB directory.

C: -> Program Files -> MongoDB -> Server -> 4.0(version) -> bin

In the bin directory, you’ll find an interesting couple of executable files.

* mongod
* mongo

Let’s talk about these two files.

mongod stands for “Mongo Daemon”. mongod is a background process used by MongoDB. The main purpose of mongod is to manage all the MongoDB server tasks. For instance, accepting requests, responding to client, and memory management.

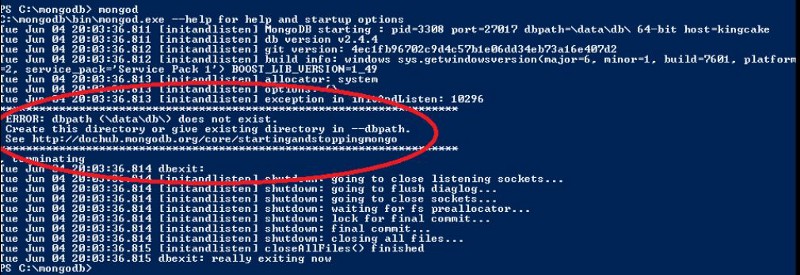
mongo is a command line shell that can interact with the client (for example, system administrators and developers).

Now let’s see how we can get this server up and running. To do that on Windows, first you need to create a couple of directories in your C drive. Open up your command prompt inside your C drive and do the following:

C:\> mkdir data/dbC:\> cd dataC:\> mkdir db

The purpose of these directories is MongoDB requires a folder to store all data. MongoDB’s default data directory path is /data/db on the drive. Therefore, it is necessary that we provide those directories like so.

If you start the MongoDB server without those directories, you’ll probably see this following error:

trying to start mongodb server without \data\db directories

After creating those two files, head over again to the bin folder you have in your mongodb directory and open up your shell inside it. Run the following command:

mongod

Voilà! Now our MongoDB server is up and running! ?

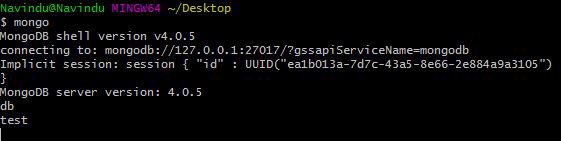
In order to work with this server, we need a mediator. So open another command window inside the bind folder and run the following command:

mongo

After running this command, navigate to the shell which we ran mongod command (which is our server). You’ll see a ‘connection accepted’ message at the end. That means our installation and configuration is successful!

Just simply run in the mongo shell:

db

initially you have a db called ‘test’

#### **Setting up Environment Variables**

To save time, you can set up your environment variables. In Windows, this is done by following the menus below:

Advanced System Settings -> Environment Variables -> Path(Under System Variables) -> Edit

Simply copy the path of our bin folder and hit OK! In my case it’s C:\Program Files\MongoDB\Server\4.0\bin

Now you’re all set!

#### **Working with MongoDB**

There’s a bunch of GUIs (Graphical User Interface) to work with MongoDB server such as MongoDB Compass, Studio 3T and so on.

They provide a graphical interface so you can easily work with your database and perform queries instead of using a shell and typing queries manually.

But in this article we’ll be using command prompt to do our work.

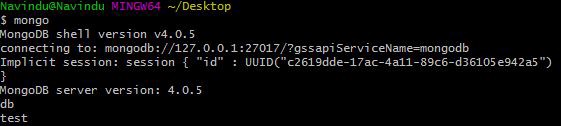
Now it’s time for us to dive into MongoDB commands that’ll help you to use with your future projects.

1. Open up your command prompt and type mongodto start the MongoDB server.

2. Open up another shell and type mongoto connect to MongoDB database server.

#### **1. Finding the current database you’re in**

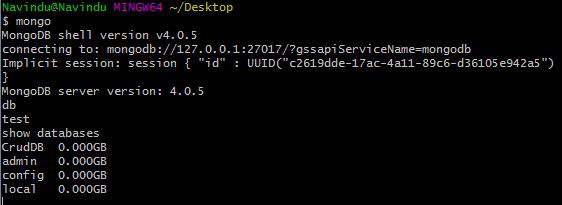
db



This command will show the current database you are in. test is the initial database that comes by default.

#### **2. Listing databases**

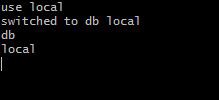
show databases



I currently have four databases. They are: CrudDB, admin, config and local.

#### **3. Go to a particular database**

use <your\_db\_name>



Here I’ve moved to the local database. You can check this if you try the command dbto print out the current database name.

#### **4. Creating a Database**

With RDBMS (Relational Database Management Systems) we have Databases, Tables, Rows and Columns.

But in NoSQL databases, such as MongoDB, data is stored in BSON format (a binary version of JSON). They are stored in structures called “collections”.

Graphical user interface, text

Description automatically generated

#### **5. For creating collection**

MongoDB Enterprise > db.automation.insert({"productName":"Iphone","Quntity":3000})

WriteResult({ "nInserted" : 1 })

#### **6.To check database:**

MongoDB Enterprise > db.automation.find()

{ "\_id" : ObjectId("6037b260109e8be63dfed112"), "productName" : "Iphone", "Quntity" : 3000 }