### **NoSQL in MongoDB Compass**

The practice purpose is to become familiar with **MongoDB Compass** tool. In this practice you are required to explore data and write some queries to retrieve data from a NoSQL database based on MongoDB.

## **1. Problem specifications**

The database contains Car Sharing information divided into two main collections: Bookings and Parkings. The most relevant information for each collection is shown in Table 1 (Parkings) and 2 (Bookings).

Name	Туре	Description	
_id	objectid	Document identifier	
address	string	Parking address of the vehicle	
city	string	City location of the vehicle	
engineType	string	Identifier of the engine type of the vehicle	
exterior	string	String describing the external condition of the vehicle during the parking	
final_date	date	Date and hour of the end of the parking period	
fuel	int32	Fuel level (0-100) during the parking period	
init_date	date	Date and hour of the beginning of the parking period	
interior	string	String describing the internal condition of the vehicle during the parking	
Іос	coordinates	Coordinates of the parking location	
plate	int32	Identifier of the vehicle's plate	
smartphoneRequired	Boolean	Boolean value denoting if the smartphone is required to start/finish the parking	
vendor	string	Company owner of the vehicle	
vin	string	Identifier of the chassis of the vehicle	

Table 1: **Parkings** database info.

Name	Туре	Description			
_id	objectid	Document id	Document identifier		
car_name	string	Vehicle's mo	Vehicle's model		
city	string	City location	where t	the vehicle has been booked	
distance	int32	Distance cov	Distance covered during the vehicle renting		
driving	object				
		distance	int32	Distance covered during the vehicle renting (in meters)	
		duration	int32	Duration of the renting (in seconds)	
engineType	string	Identifier of	Identifier of the engine type of the vehicle		
exterior	string		String describing the external condition of the vehicle during the renting		
final_address	string	Address of th	Address of the final position of the renting period		
final_date	date	Date and ho	Date and hour of the end of the renting period		
final_fuel	int32	Fuel level (0-100) at the end of the renting period			
init_address	int32	Address of the starting position of the renting period			
init_date	date	Date and hour of the beginning of the renting period			
init_fuel	int32	Fuel level (0-100) at the beginning of the renting period			
interior	string	String describing the internal condition of the vehicle during the renting			
plate	int32	Identifier of the vehicle's plate			
smartphoneRequired	Boolean	Boolean value denoting if the smartphone is required to start/finish the parking			
vendor	string	Company owner of the vehicle			
walking	object				
		distance	int32	Walk distance to reach the vehicle (in meters).	
		duration	int32	Duration of the walking trip to reach the vehicle (in	

			seconds).

Table 2: **Bookings** database info.

# 2. Database Connection

### Remote database connection

- 1. Download MongoDB Compass at https://www.mongodb.com/try/download/compass
- 2. Install and open the application.
- 3. Create a free cluster (if you do not have already one)
  - a. create a MongoDB account (https://www.mongodb.com/cloud/atlas/register)
  - b. Select "*Shared Cluster*" option (free cluster). The default settings are the ones set to get the account completely free.
  - c. give a name to your cluster
  - d. go to Database Access
    - i. click on "Add New Database User"
    - ii. select authentication method with password
    - iii. fill in the form
    - iv. leave all the default options
  - e. configure remote access
    - i. click on "*Wetwork Access"*
    - ii. click on "Add IP Address"
    - iii. enter in the Access List Entry field 0.0.0.0/0
- 4. Get string connection from MongoDB Atlas server
  - a. click on the CONNECTIONS buttons
  - b. select "Connecting using MongoDB Compass"
  - c. copy the connection string
- 5. Paste the connection string into MongoDB Compass
- 6. Click on Connect

### 3. Create a database in MongoDB Compass

#### 1. Click Create Database

- a. Assign a name to the database
- b. insert the name "Bookings" in the field "Collection Name"
- c. leave all the default values for the other options
- 2. Add a new collection
  - a. click on "Create Collection" button
  - b. enter the name "Parkings"
  - c. leave all the default values for the other options
- 3. Download collection data from the website
- 4. Add data to each collection
  - a. select one collection
    - b. click "Add Data" button, then Import file
    - c. select the file of the corresponding collection
    - d. select Json type
    - e. click Import
- 5. Select the Parkings collection
- 6. Go to the "Indexes" tab
- 7. Create a 2dsphere index on the loc field

# 4. Analyze the database using the Schema analyzer

	MongoDB Compass - cluster0.nyhqz.mongodb.net/dbdmg.bookings
Local	döding bookings Schema * +
✓ 14 DBS 33 COLLECTIONS C	dbdmg.bookings Documents 78.4k TOTAL SIZE AVG.
HOSTS cluster0-shard-00-01.nyhq cluster0-shard-00-02.nyhq cluster0-shard-00-00.nyhq	Documents Aggregations Schema Explain Plan Indexes Validation
CLUSTER Replica Set (Cluster0-shar 3 Nodes	
EDITION MongoDB 4.4.10 Enterprise	nnl
<b>Q</b> Filter your data	
> admin > config	Explore your schema
✓ dbdmg ■ bookings	Quickly visualize your schema to understand the frequency, types and ranges of fields in your data set.
□ parkings > exams	Analyze Schema
> local > restaurants	Learn more about schema analysis in Compass
> sample_airbnb	
> sample_analytics	
> sample_geospatial	
+	
>_MONGOSH	^
	MongoDB Compass - cluster0.nyhqz.mongodb.net/dbdmg.bookings
Local	dbdmg.bookings
	schema schema
✓ 14 DBS 33 COLLECTIONS C ☆ FAVORITE	dbdmg.bookings Documents 78.4k TOTAL SIZE AVG. SIZ
HOSTS cluster0-shard-00-01.nyhq	Documents Aggregations Schema Explain Plan Indexes Validation
cluster0-shard-00-02.nyhq cluster0-shard-00-00.nyhq	(Indee) { field: 'volue' }
CLUSTER Replica Set (Cluster0-shar 3 Nodes	This report is based on a sample of 1000 documents.
EDITION MongoDB 4.4.10 Enterprise	_id objectid
<b>Q</b> Filter your data	Frst: 2017-09-18 02:38:59 kest: 2017-10-01 21:59:51
> admin	
> config	car_category_id 100%
✓ dbdmg ■ bookings ····	initiz undefined sow
parkings	
> exams	100
> local	
> restaurants	car_category_type_id 100%
> sample_airbnb > sample_analytics	int32 undefined 50%
<ul> <li>sample_analytics</li> <li>sample_geospatial</li> </ul>	0% 405
+	
>_MONGOSH	^

- 1. (Bookings) Identify the most common percentage(s) of fuel level at the beginning of the renting period.
- 2. (Bookings) Identify the most common percentage(s) of fuel level at the end of the renting period.
- 3. (Parkings) Identify the time range(s) with most parking requests (start parking).
- 4. (Parkings) Identify the time range(s) with most booking requests (end parking).
- 5. (Parkings) Visualize on the map the vehicles having the fuel level lower than 5%.

# 5. Querying the database

• • •	MongoDB Compass - cluster0.nyhqz.mongodb.net/dbdmg.parkings	
Local	dbdmg.parkings Documents	3
✓ 14 DBS 33 COLLECTIONS C ☆ FAVORITE	dbdmg.parkings DOCUMENTS 78.6k	INDEXES 1 TOTAL SIZE AVG. SIZE AVG. SIZE AVG. SIZE 1.7MB 1.7MB
HOSTS cluster0-shard-00-01.nyhq cluster0-shard-00-02.nyhq cluster0-shard-00-00.nyhq	Documents Aggregations Schema Explain Plan Indexes Validation	・ OPTIONS FIND RESET つ …
CLUSTER Replica Set (Cluster0-shar	0 PROJECT         {plate:1, _id:0}           0 SORT) { field: -1 } or [['field', -1]]         0 MAX TIT	NE MS) 60000
3 Nodes	(COLLATION { locale: 'simple' }	0
EDITION MongoDB 4.4.10 Enterprise	▲ VIEW	cuments 1 - 20 of 78557 < > C REFRESH
<b>Q</b> Filter your data	plate: "550/FK213PV"	প্র
> admin		
> config ∽ dbdmg	plate: "540/FK820LX"	
bookings	plate: "EZ055TY"	
parkings     ····     exams     local	plate: "EZ283GW"	
<ul> <li>restaurants</li> <li>sample_airbnb</li> </ul>	plate: "EZ270GW"	
<ul> <li>sample_analytics</li> <li>sample_geospatial</li> </ul>	plate: "311/FF2865J"	
	plate: "196/FF350NT"	
>_MONGOSH		

(Parkings) Find the plates and the parking addresses of the vehicles that begin the booking (end parking) after 2017-09-30 at 6AM.

**Hint**: it is possible to use the function Date("<YYYY-mm-ddTHH:MM:ss>")

- 2. (Parkings) Find the addresses and the level of fuel of the vehicles that during the parking period had at least 70% of fuel level. Order the results according to descending value of fuel level.
- 3. (Parkings) Find the plate, the engine type and fuel level for `car2go' vehicles (vendor) with good internal and external conditions.
- 4. (Bookings) For the renting that required a walking distance greater than 15 Km (to reach the vehicle), find the hour and the fuel level at the beginning of the renting period. Order results according to decreasing initial fuel level.

## 6. Data Aggregation

- 5. (Bookings) Group documents according to their fuel level at the end of the renting. For each group, select the average fuel level at the beginning of the renting period.
- 6. (Bookings) Select the average driving distance for each vendor. On average, for which vendor the users cover longer distances?

# 7. Bonus Queries

7. (Parkings) Find the vehicles parked less than a mile far from Piazza San Carlo (coordinates: 7.683016, 45.067764).

**Hint:** use the operators <u>\$geoWithin</u> and <u>\$centerSphere</u>.

8. (Parkings) Repeat the query at the previous step using the coordinates of a place of personal interest in Turin (e.g. Politecnico di Torino) using Open Street Maps to find the exact coordinates (www.openstreetmap.org, inverse the coordinates order).