



## Data Management and Visualization

Politecnico di Torino

### NoSQL in MongoDB Compass – Practice 4

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

Table 1: **Parkings** database info.

Name	Type	Description		
<b>_id</b>	objectid	Document identifier.		
<b>car_name</b>	string	Vehicle's model		
<b>city</b>	string	City location where the vehicle has been booked.		
<b>distance</b>	int32	Distance covered during the vehicle renting.		
<b>driving</b>	object	<b>distance</b>	int32	Distance covered during the vehicle renting (in meters).
		<b>duration</b>	int32	Duration of the renting (in seconds)
<b>engineType</b>	string	Identifier of the engine type of the vehicle.		
<b>exterior</b>	string	String describing the external condition of the vehicle during the renting.		
<b>final_address</b>	string	Address of the final position of the renting period.		
<b>final_date</b>	date	Date and hour of the end of the renting period.		
<b>final_fuel</b>	int32	Fuel level (0-100) at the end of the renting period.		
<b>init_address</b>	int32	Address of the starting position of the renting period.		
<b>init_date</b>	date	Date and hour of the beginning of the renting period.		
<b>init_fuel</b>	int32	Fuel level (0-100) at the beginning of the renting period.		
<b>interior</b>	string	String describing the internal condition of the vehicle during the renting.		
<b>plate</b>	int32	Identifier of the vehicle's plate.		
<b>smartphoneRequired</b>	Boolean	Boolean value denoting if the smartphone is required to start/finish the parking.		
<b>vendor</b>	string	Company owner of the vehicle.		
<b>walking</b>	object	<b>distance</b>	int32	Walk distance to reach the vehicle (in meters).
		<b>duration</b>	int32	Duration of the walking trip to reach the vehicle (in seconds).

Table 2: **Bookings** database info.

## 2. Database Connection

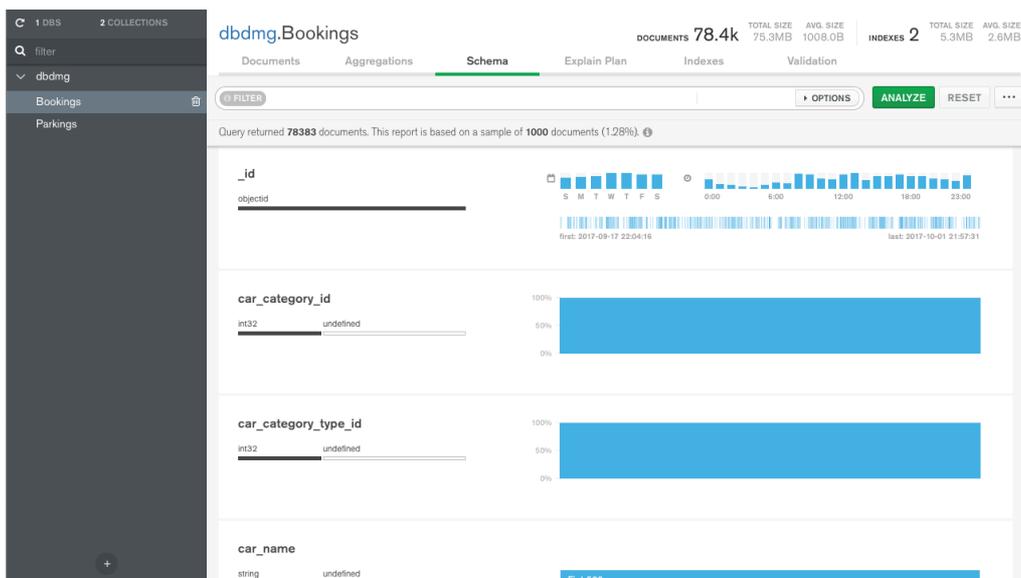
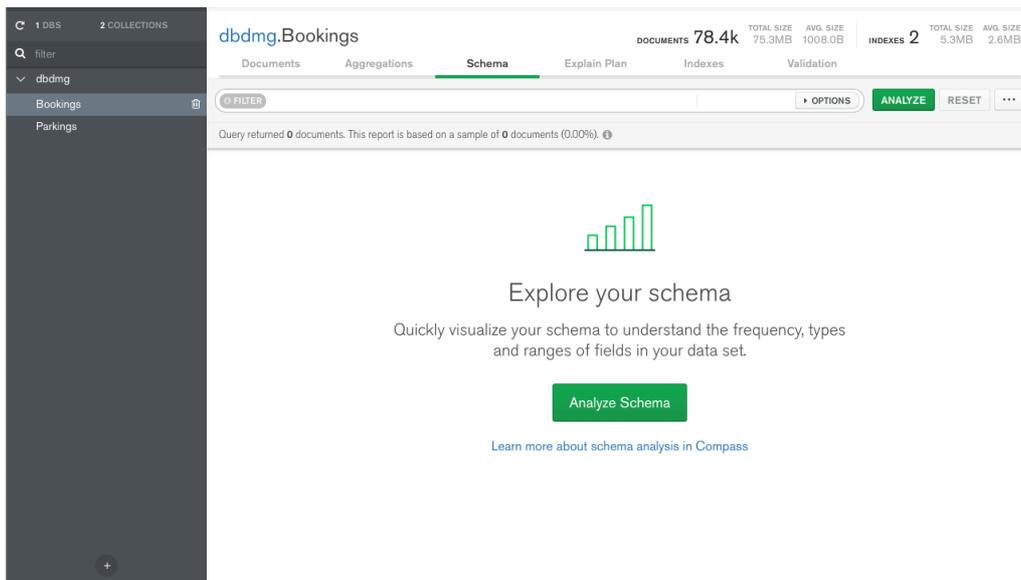
### Remote database connection

1. Download MongoDB Compass using one of the following links:
  - Ubuntu (.deb): [https://downloads.mongodb.com/compass/mongodb-compass\\_1.18.0\\_amd64.deb](https://downloads.mongodb.com/compass/mongodb-compass_1.18.0_amd64.deb)
  - RedHat (.rpm): [https://downloads.mongodb.com/compass/mongodb-compass-1.18.0.x86\\_64.rpm](https://downloads.mongodb.com/compass/mongodb-compass-1.18.0.x86_64.rpm)
  - Windows (.exe) 64 bit: <https://downloads.mongodb.com/compass/mongodb-compass-1.18.0-win32-x64.exe>
  - Mac OS (.dmg): <https://downloads.mongodb.com/compass/mongodb-compass-1.18.0-darwin-x64.dmg>

In LABINF you need to login from Windows and download the adequate version of MongoDB Compass

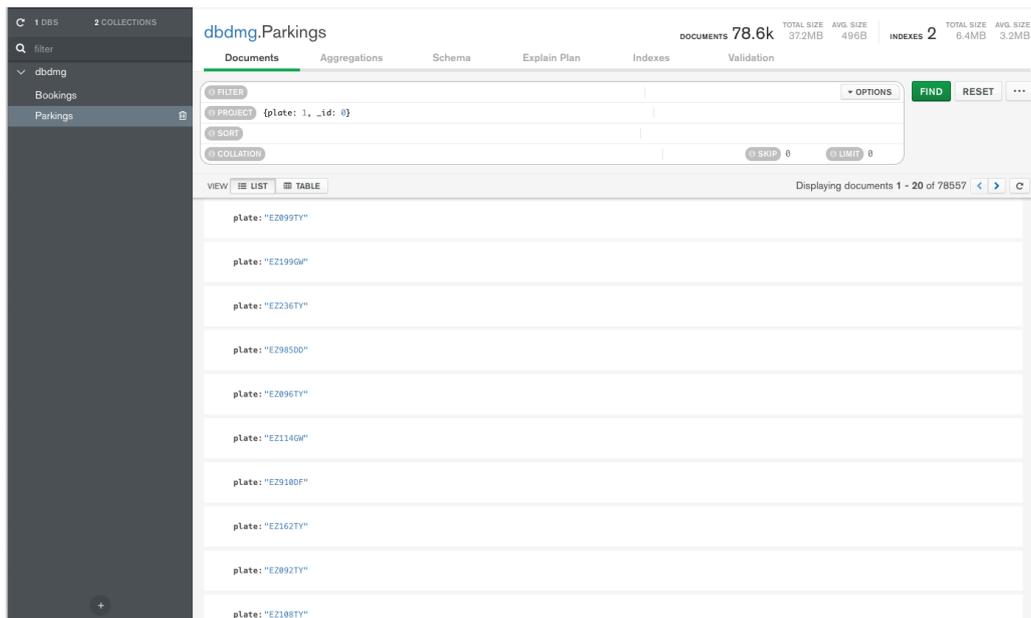
2. Install and open the application.
3. Connect to the remote database using the following connection parameters:
  - a. **Hostname:** bigdatadb.polito.it
  - b. **Port:** 27017
  - c. **Authentication:** Username/Password
  - d. **Username:** Compass
  - e. **Password:** Compass19!
  - f. **Authentication database:** dbdmg
  - g. **SSL:** Unvalidated (insecure)
4. (Optional) Specify a **Favourite Name** to easily connect to the database in the future. Click on **Save Favourite**.
5. Click on **Connect**.
6. Access to **dbdmg**.
7. Access to a specific partition **Parkings/Bookings**.

### 3. Analyze the database using the *Schema analyzer*



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%.

## 4. Querying the database



1. (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( "<math>\langle YYYY-mm-ddTHH:MM:ss \rangle"</math> )` )
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.

## 5. 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?

## 6. 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 [\\$geoWithin](#) and [\\$centerSphere](#).
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](http://www.openstreetmap.org), inverse the coordinates order).