

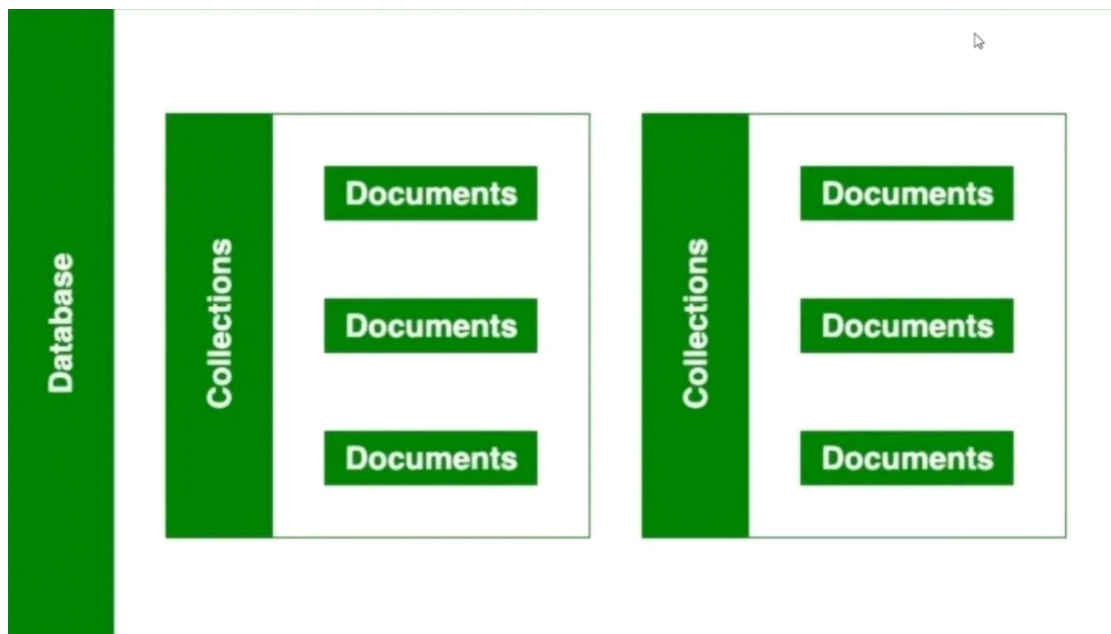
Corso MongoDB

Lezione 1 – Introduzione a MongoDB

Cosa è?

MongoDB (da "humongous", enorme) è un **DBMS non relazionale**, orientato ai **documenti**. Classificato come un **database** di tipo **NoSQL**, MongoDB si allontana dalla struttura tradizionale basata su tabelle dei database relazionali in favore di documenti in stile JSON con schema dinamico (MongoDB chiama il formato BSON), rendendo l'integrazione di dati di alcuni tipi di applicazioni più facile e veloce.

Struttura differente rispetto ai DB relazionali



"Tabella sostituita dalla Collection"

Database SQL:

DB

|

+--- Tabella

|

+--- Colonna

|

+--- Riga

Database NoSQL:

DB

|

+--- Collection

|

+--- Document

|

+--- Riga

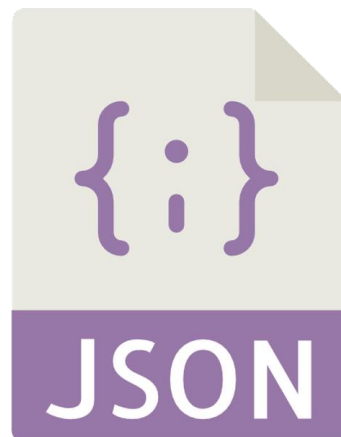
Es. Collection = Utente, Document = Singolo Utente

Il Document ha una struttura simil JSON, ma denominata BSON

Es. Document BSON:

```
{  
  name: "John",  
  age: "32",  
  hobbies: ["Sport", "D&D", "Bowling"]  
}
```

BSON \neq JSON



La struttura è però fondamentalmente simile!

MA...lo schema dei documents non è statico, come nel caso dei DB SQL, dove ogni tabella deve avere righe e colonne che devono mantenere la medesima struttura, qui, ogni document di una collection, può essere differente dall'altro, quindi più **dinamico**.

Es.: Document 1 : name, age, hobbies --- Document 2 : name

Lezione 2 – Installazione MongoDB

- 1) Download della versione community server (free version)
- 2) Programmi > MongoDB > Server > 5.0(v.) > bin > *

*

- ⑩ **mongod** → Server
- ⑩ **mongo** → shell dei comandi

N.B.: La shell del Server potrebbe rilevare un problema, ovvero la (mancanza di cartella data > db)

Risoluzione: andare in disco C, creare una nuova cartella data ed inserire al suo interno una nuova cartella db.

Comandi principali:

Show dbs : mostra database presenti.

use nome_db : creerà un nuovo database.

N.B. : Pur avendo creato un nuovo DB, questo non verrà mostrato se privo di dati internamente.

Lezione 3 – Inserimento dati

Creazione di una Collection:

Sintassi : db.nome_collection

(es.: db.utenti)

Creazione di un Document:

Sintassi : db.nome_collection.insertOne({json})

es.: db.utenti.insertOne({"nome":"Luca", "cognome":"Rossi", "eta": 25, "citta":"Milano"})

// Darà come output:

```
{
  "acknowledged" : true,
  "insertedId" : ObjectId("61ffcafe424952c90e6cf7e")
}
```

// Con un **id univoco autogenerato** e la presa in carica dell'inserimento

Creazione di più Document:

Sintassi : db.nome_collection.insertMany([json1, json2, json3, json4, json5])

```
db.utenti.insertMany([{"nome":"Jonah", "cognome":"Pavlov", "eta": 20,
"citta":"Poggibonsi"}, {"nome":"Giovanni", "cognome":"Storti", "eta": 35, "citta":"Genova"}])
```

// Output positivo

```
{
  "acknowledged" : true,
  "insertedIds" : [
    ObjectId("61ffff09e424952c90e6cf7f"),
    ObjectId("61ffff09e424952c90e6cf80")
  ]
}
```

Lezione 4 – Selezionare e filtrare dati

Selezionare tutti i documents di una collection:

Sintassi : db.nome_document.find()

Sintassi : db.nome_document.find().pretty : darà una visualizzazione migliore del document

(es.: db.utenti.find().pretty())

// Output desiderato:

```
{
  "_id" : ObjectId("61fffcfe424952c90e6cf7e"),
  "nome" : "Luca",
  "cognome" : "Rossi",
  "eta" : 25,
  "citta" : "Milano"
}
{
```

```

    "_id" : ObjectId("61ffff09e424952c90e6cf7f"),
    "nome" : "Jonah",
    "cognome" : "Pavlov",
    "eta" : 20,
    "citta" : "Poggibonsi"
  }
{
  "_id" : ObjectId("61ffff09e424952c90e6cf80"),
  "nome" : "Giovanni",
  "cognome" : "Storti",
  "eta" : 35,
  "citta" : "Genova"
}

```

Selezionare un solo document di una collection:

Sintassi : `db.nome_document.findOne({citta: "Milano"})`

(es.: `db.utenti.findOne({citta: "Milano"})`)

// Cercherà un document con valore di proprietà citta uguale alla stringa "Milano", ma **soltanto uno**.

Operatori di comparazione:

Name	Description
<code>\$eq</code>	Matches values that are equal to a specified value.
<code>\$gt</code>	Matches values that are greater than a specified value.
<code>\$gte</code>	Matches values that are greater than or equal to a specified value.
<code>\$in</code>	Matches any of the values specified in an array.
<code>\$lt</code>	Matches values that are less than a specified value.
<code>\$lte</code>	Matches values that are less than or equal to a specified value.
<code>\$ne</code>	Matches all values that are not equal to a specified value.
<code>\$nin</code>	Matches none of the values specified in an array.

"dalla documentazione"

Selezionare tutti i documents con un parametro superiore/inferiore a x :

Sintassi : db.nome_document.find({key: {\$gt: x}})

(es.: db.utenti.find({eta: {\$gt:30}}))

// Output: troverà tutti gli utenti con età superiore a 30

```
{ "_id" : ObjectId("61ffff09e424952c90e6cf80"), "nome" : "Giovanni", "cognome" : "Storti", "eta" : 35, "citta" : "Genova" }
```

Esempio ad uso di un **operatore logico (AND)** in una **comparazione**

db.utenti.find({\$and:[{eta: {\$gt: 19}},{citta:"Milano"}]})

// Output: Cercherà i document nelle quali la citta è uguale a Milano e l'eta è superiore a 19 anni

Lezione 5 – Modifica/aggiornamento di dati in un document

document – filtro – modifica effettiva

Selezione e modifica di un document:

Sintassi: db.nome_document.updateOne({field:"value"}, {\$set: {field: newValue, field2: newValue2}})

(es.: db.utenti.updateOne({nome:"Luca"}, {\$set: {eta: 29, cognome: "Poplite"}}))

// Output positivo:

```
{ "acknowledged" : true, "matchedCount" : 0, "modifiedCount" : 0 }
```

Selezione e modifica di più documents:

Sintassi: db.nome_document.updateMany({field:"value"}, {\$set: {field: newValue}})

(es.: db.utenti.updateMany({citta:"Milano"}, {\$set: {cap: "00476"}}))

// Output : modificherà, aggoingendo il field cap a coloro che hanno come città, nel filtro, Milano.

// Output positivo:

```
{ "acknowledged" : true, "matchedCount" : n, "modifiedCount" : n }
```

Selezione e modifica di tutti documents usando l'operatore incremento (\$inc):

Sintassi: db.nome_document.updateMany({}, {\$inc: {field:1}})

(es.: db.utenti.updateMany({}, {\$inc: {eta:1}}))

// Output : modificherà l'eta di tutti i documents incrementando il campo di 1

// Output positivo:

```
{ "acknowledged" : true, "matchedCount" : 3, "modifiedCount" : 3 }
```

Upsert: esegui un operazione di modifica, se il filtro non trova un item che corrisponde al filtro, crea un nuovo item

Sintassi:

db.nome_document.updateMany({fiels: value doesn't exist}, {\$set: {field1: value1, field2: value2}}, {upsert:true})

(es.: db.utenti.updateMany({nome:"Massimo"}, {\$set: {cognome: "Blu, eta:23", citta:"Roma"}}, {upsert:true}))

// Output positivo:

```
{
  "acknowledged" : true,
  "matchedCount" : 0,
  "modifiedCount" : 0,
  "upsertedId" : ObjectId("6200102e060989d9d3785f54")
}
```

Lezione 6 – Eliminazione di dati in un document

Rimozione di un document :

Sintassi: db.nome_document.deleteOne({field:value})

(es.: db.utenti.deleteOne({nome:"Massimo"}))

// Output esito positivo:

```
{ "acknowledged" : true, "deletedCount" : 1 }
```

Rimozione di un document con delle condizioni:

Sintassi: db.nome_document.deleteOne({\$and: [{field: {\$gt:35}}, {field2: value2}]})

Es. db.utenti.deleteOne({\$and: [{eta: {\$gt:35}}, {citta: "Genova"}]})

// Eliminerà l'elemento con citta Genova e età superiore a 35

// Output positivo:

```
{ "acknowledged" : true, "deletedCount" : 1 }
```

Rimozioni più documents:

Sintassi: db.nome_document.deleteMany({field: value})

Es.: db.utenti.deleteMany({citta: "Milano"})

// Output: Eliminerà tutti i document con citta uguale a Milano

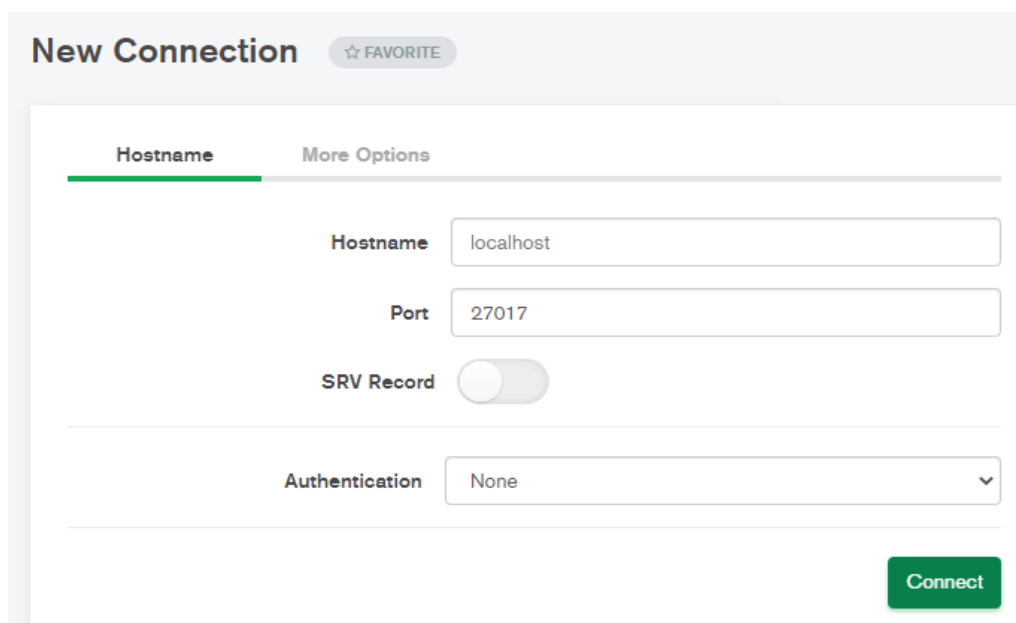
Lezione 6 – Uso di MongoDB Compass

Cosa è ?

MongoDB Compass è l'alternativa **UI** di comando alla shell di MongoDB

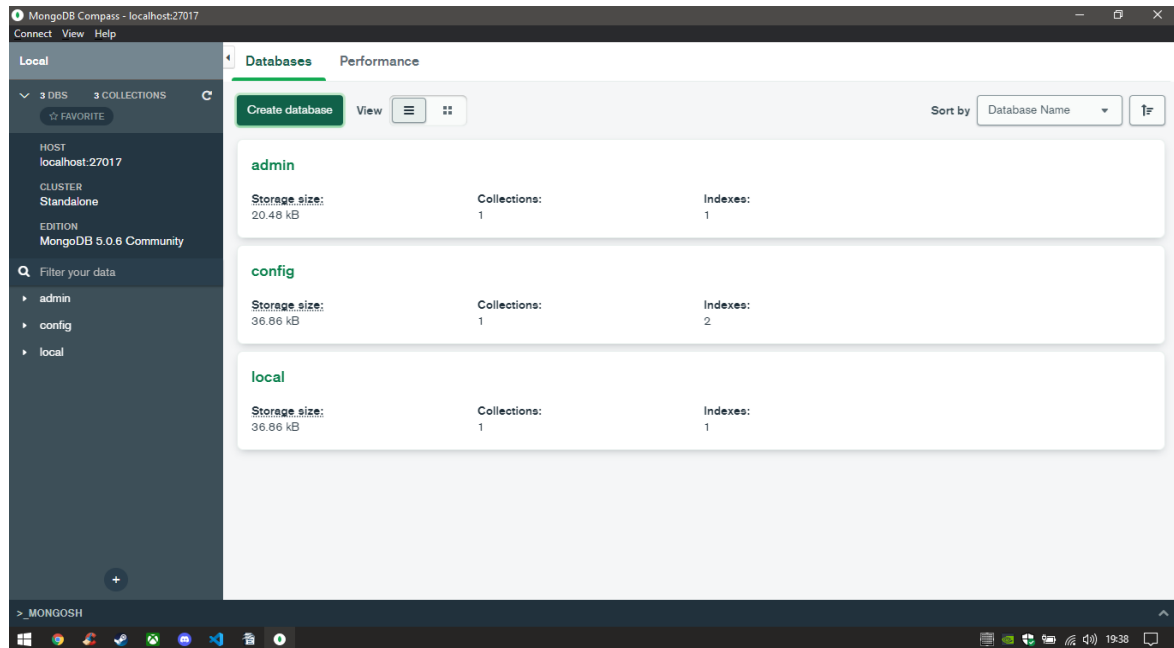
Esistono però molte altre UI utili per gestire il DB

Connettiamo al DB --> [Fill in connection fields individually](#)

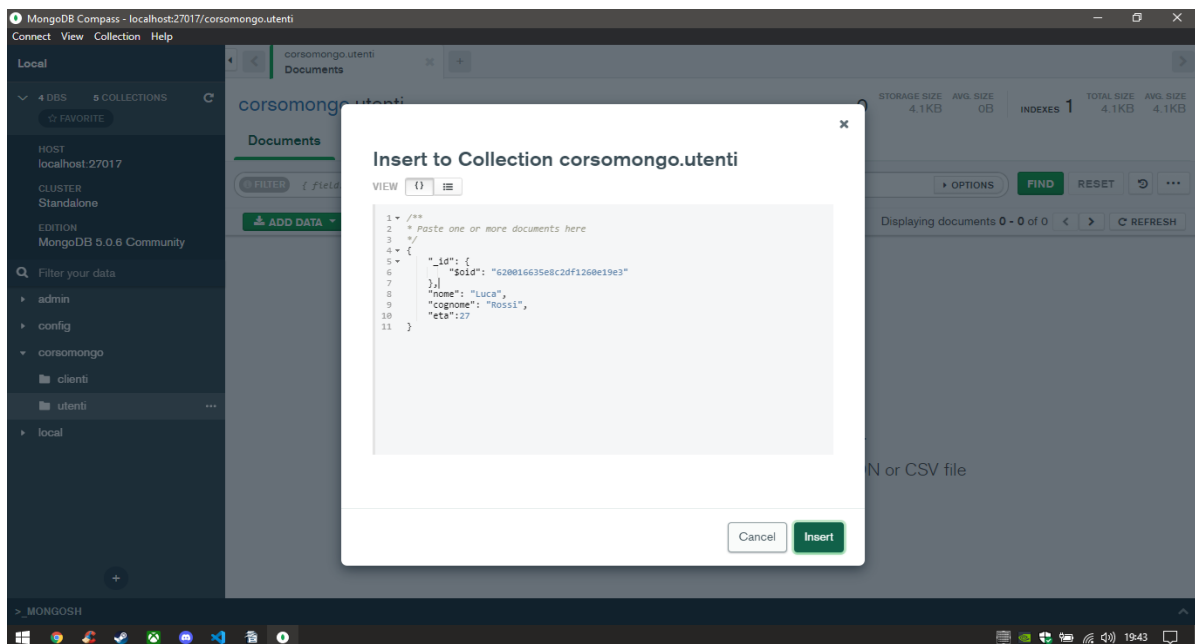


The screenshot shows the 'New Connection' dialog in MongoDB Compass. At the top, there's a 'New Connection' title and a 'FAVORITE' button. Below this, there are two tabs: 'Hostname' (selected) and 'More Options'. The 'Hostname' tab contains three input fields: 'Hostname' with the value 'localhost', 'Port' with the value '27017', and 'SRV Record' which is a toggle switch currently turned off. Below these fields is an 'Authentication' dropdown menu set to 'None'. At the bottom right of the dialog is a green 'Connect' button.

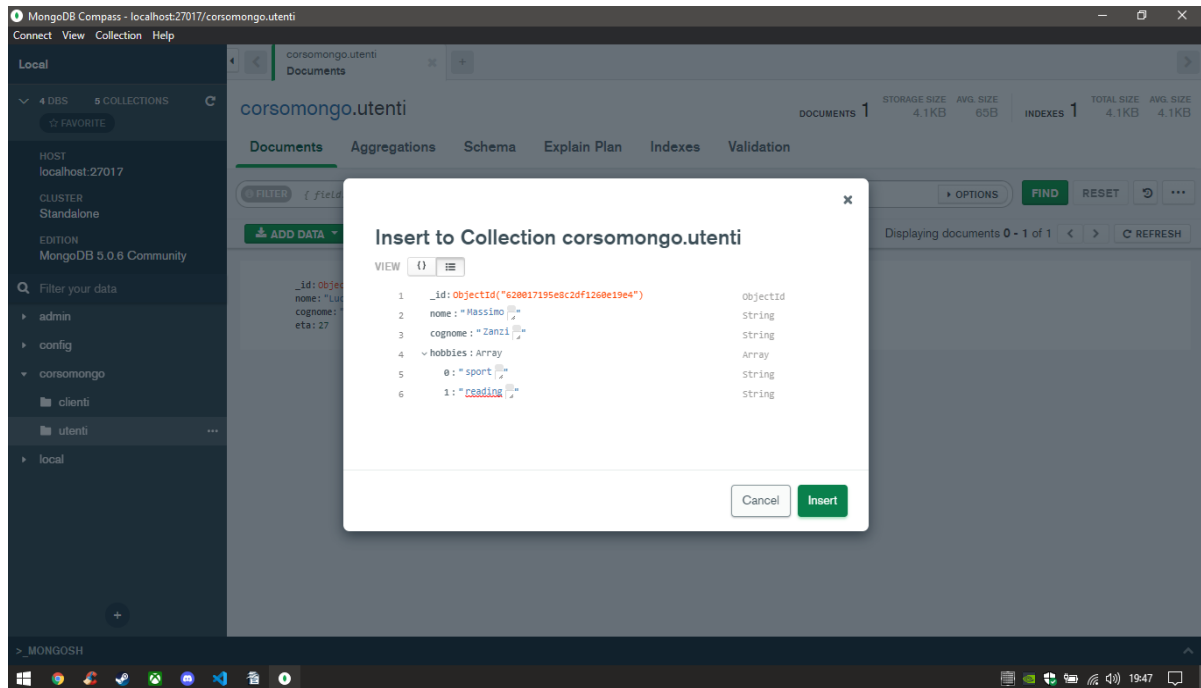
L'interfaccia è molto intuitiva e permette la rimozione, creazione e modifica di records, documents.
Collections e interi databases



Esempio Insert Document



Esempio insert con altro metodo



Naturalmente sono possibili tutte le operazioni quali modifica, duplicazione di un document, find, ecc...

Note:

- Link Corso https://www.youtube.com/playlist?list=PLP5MAKLy8IP_QKLouwhi-cKC_nDhYoPfR
- Documentazione MongoDB: <https://docs.mongodb.com/>
- Incollare testo in Mongo Shell --> Tasto dx
- Ricorda sempre di entrare nel database usando use nome_db