

# MongoDB

---

A DOCUMENT-ORIENTED DATABASE

# Getting started

---

# Introduzione

---

I database document-oriented sostituiscono il concetto di *riga* con un modello più flessibile: il **documento**

- Possibilità di rappresentare relazioni **gerarchiche complesse in un unico documento**
- **Non esiste uno schema predefinito**

Alcune delle caratteristiche principali:

- **Tanti indici a disposizione: composti, geo-spaziali, full-text**
- Un meccanismo di **aggregation pipeline** per costruire aggregazioni complesse attraverso la concatenazione di piccoli «pezzi»
- **Diversi tipi di collezione: time-to-live, fixed-size**
- Possibilità di usare **script nel linguaggio Javascript** per manipolare i dati

# Introduzione

---

Alcuni concetti di base:

- Il **documento** è l'unità di base, più espressiva della sua controparte relazionale (i.e., la *riga*)
- La **collezione** è la controparte documentale del concetto di *tabella*; diversamente da quest'ultima, non prevede uno schema di base
- Un'istanza di MongoDB può contenere diversi **database**, ognuno con le sue collezioni
- In ogni documento viene automaticamente inserito un campo speciale, identificato dalla chiave **\_id**, il cui valore è unico all'interno della collezione (corrisponde alla chiave primaria)

# Documenti

---

I documenti corrispondono sostanzialmente a oggetti JSON, ma più espressivi

- E' possibile utilizzare tipi di dato che il formalismo JSON non prevede

In generale, sono **ricorsivamente** definiti **come oggetti composti da coppie ordinate chiave-valore**, in cui:

- La *chiave* è una stringa case-sensitive
  - Non si possono usare i caratteri “.” e “\$”
  - Non possono esistere due chiavi identiche all'interno dello stesso oggetto
- Il *valore* può essere di diversi tipi:
  - Un *tipo semplice* (e.g., stringa, numero, data, ecc.)
  - Un altro *oggetto*
  - Un *array* di valori
- Generalmente, l'ordine delle chiavi non è importante

# Documenti

---

## Un esempio

```
{
  "_id": ObjectId("5037ee4a1084eb3ffeed7228"),
  "info": {
    "nome": "Enrico",
    "dataNascita" : ISODate("1988-08-04T20:42:00.000Z")
  },
  "interessi": ["Calcetto", "Viaggi", "Serie TV"],
  "didattica": [{
    "corso": "Big Data",
    "datore": "Università di Bologna"
  }, {
    "corso": "Introduzione ai sistemi di basi di dati",
    "datore": "FITSTIC"
  }]
}
```

# Collezioni

---

Una collezione è costituita da un insieme di documenti

Non esiste uno schema di base

Allora perché creare più collezioni invece che tenerne una sola?

- Comodità
- Performance
- Data locality
- Indici diversi in collezioni diverse

Una collezione è identificata da un nome

- Non si può usare il carattere “\$”, ma si può usare il “.”, in particolare per organizzare concettualmente le collezioni in sotto-collezioni
  - E.g., blog.posts, blog.authors, ec..

# Database

---

Un'istanza di MongoDB può contenere tanti database, ciascuno dei quali può ospitare tante collezioni

Ogni database ha il suo meccanismo di gestione dei permessi ed è salvato in un file dedicato

- Di norma si utilizza un database per ogni applicazione

I database sono identificati da un nome

- Ci sono molte restrizioni sui caratteri (usare caratteri alfanumerici ASCII)



# Database

---

Alcuni database sono riservati

## admin

- E' il database principale in termini di **autenticazione**; gli utenti assegnati a questo database possono accedere anche a tutti gli altri
- Alcuni comandi possono essere eseguiti solo da questo database (e.g., elencare tutti i database, spegnere il server)

## local

- In un cluster, ne esiste **uno per ogni macchina** in cui è installato MongoDB
- Può essere usato per memorizzare **dati che non devono essere distribuiti**

## config

- Memorizza informazioni utili per l'utilizzo in modalità distribuita

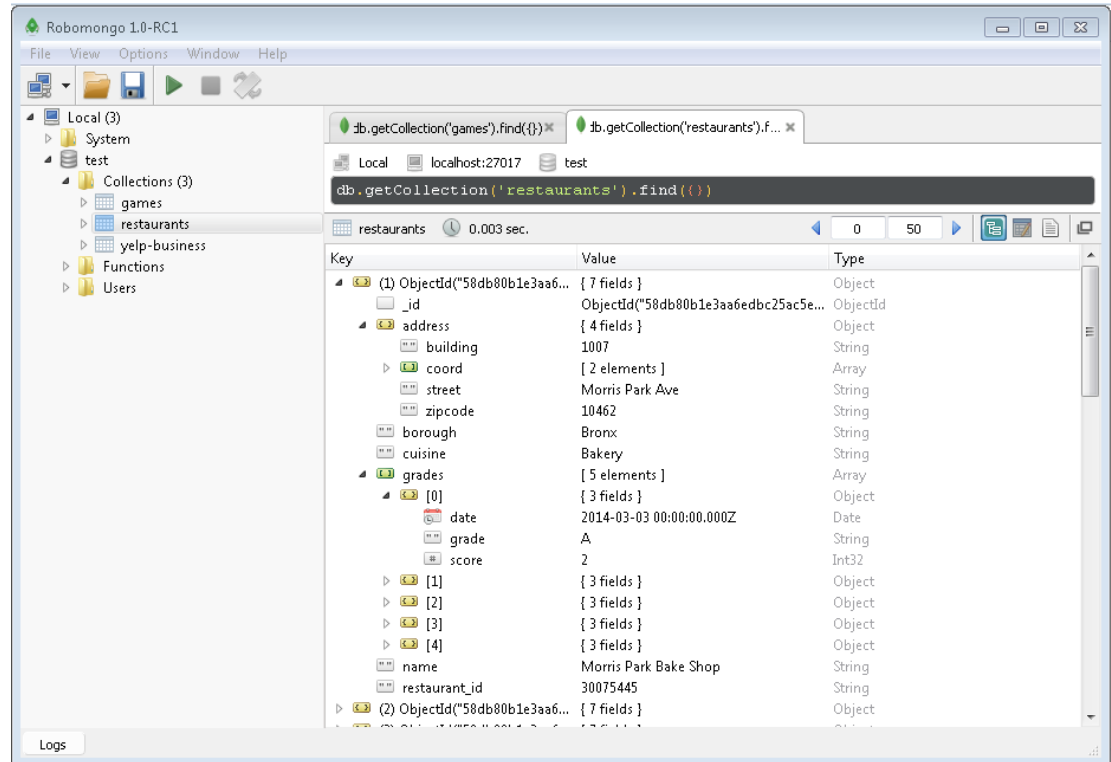
# Connessione via Robo3T

## Perché Robo3T?

- Semplifica la gestione e la navigazione del database
- Incorpora una shell di MongoDB

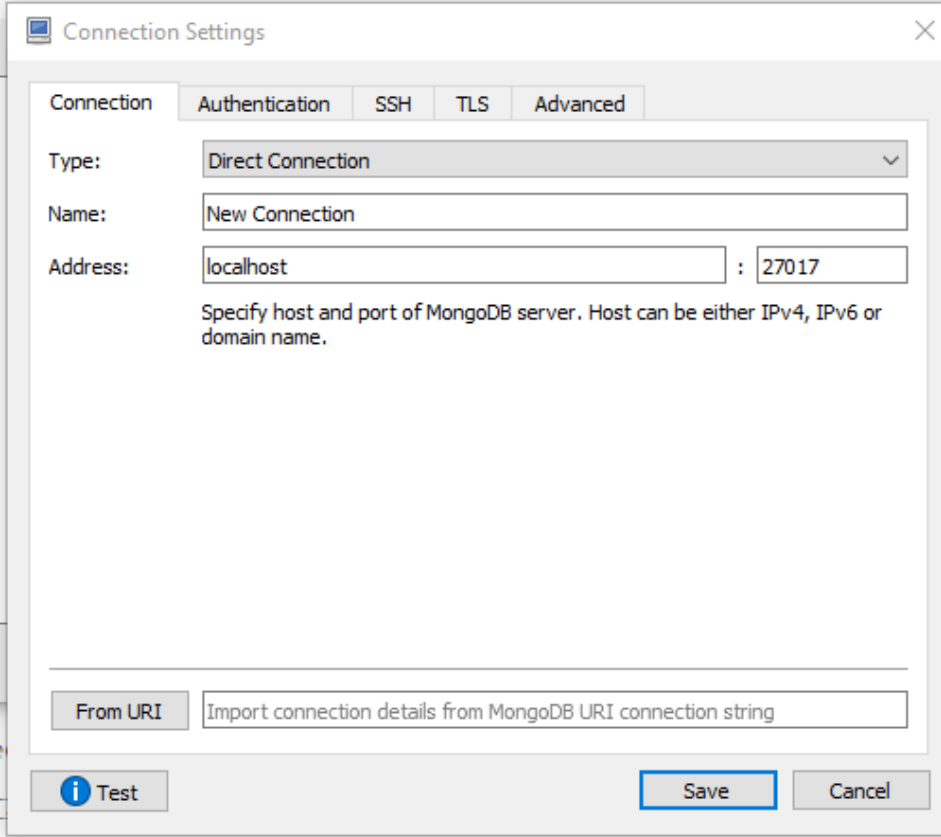
## Parametri:

- Connessione a localhost
- Porta 27017



Robo3T è la nuova versione di Robomongo

# Connessione via Robo3T



The screenshot shows the 'Connection Settings' dialog box in Robo3T. The 'Connection' tab is selected, showing fields for 'Type' (Direct Connection), 'Name' (New Connection), and 'Address' (localhost : 27017). A note below the address field states: 'Specify host and port of MongoDB server. Host can be either IPv4, IPv6 or domain name.' At the bottom, there is a 'From URI' button and a text box for importing connection details from a MongoDB URI. The 'Test', 'Save', and 'Cancel' buttons are at the bottom right.

Connection Settings

Connection Authentication SSH TLS Advanced

Type: Direct Connection

Name: New Connection

Address: localhost : 27017

Specify host and port of MongoDB server. Host can be either IPv4, IPv6 or domain name.

From URI Import connection details from MongoDB URI connection string

Test Save Cancel

# Database per le esercitazioni

---

## Ristoranti

- <https://raw.githubusercontent.com/mongodb/docs-assets/primer-dataset/primer-dataset.json>
- 25359 documenti relativi a ristoranti (nome, indirizzo, tipo di cucina, voti)

## Partite NBA

- <http://bit.ly/1gAatZK>
- 31686 documenti relativi a 30 anni di partite dell'NBA (data, rose, statistiche)

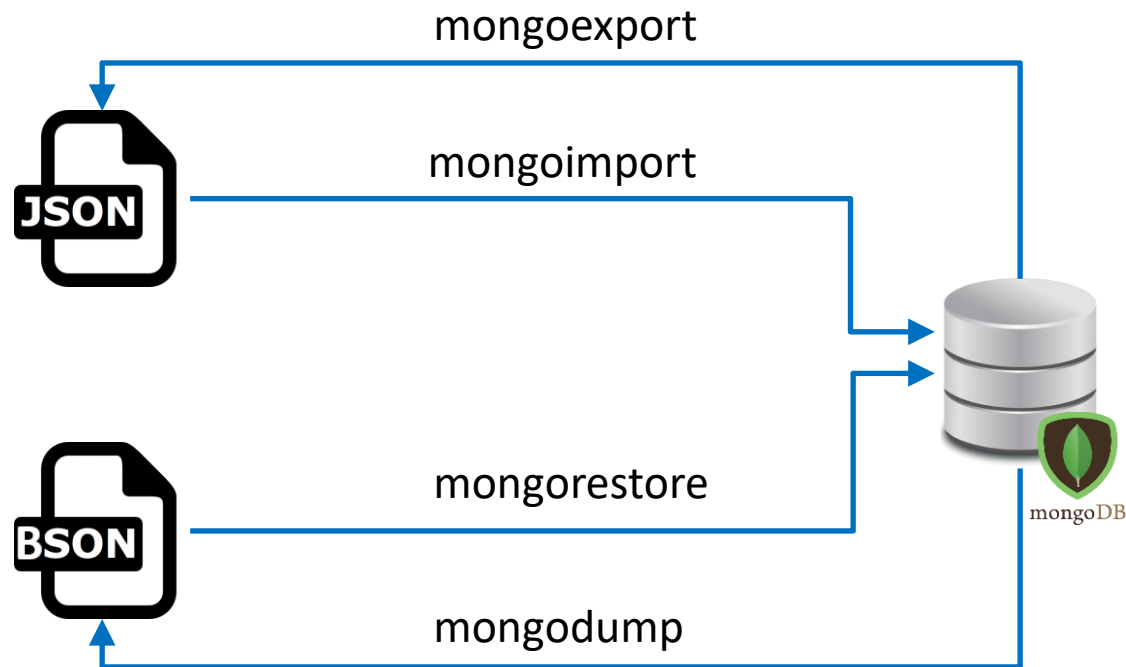
## Yelp

- [https://www.yelp.com/dataset\\_challenge](https://www.yelp.com/dataset_challenge)
- Dati reali messi a disposizione della ricerca scientifica
  - Più di 50.000\$ distribuiti in competizioni, più di 100 paper accademici

```
mongorestore --collection games --db test C:\games.bson
```

# Strumenti di import/export

---



# Comandi di base

---

La maggior parte dei comandi di MongoDB sono metodi dell'oggetto `db`

Di base, la shell si collega al database vuoto *test*

- `db` – mostra il nome del database
- `db. + tab` – mostra i metodi richiamabili

## Alcuni esempi

- `db.getMongo().getDBs()` – mostra i database presenti nell'istanza
- `db.getCollectionNames()` – mostra i nomi delle collezioni nel DB corrente
- `db.getSisterDB("foo")` – passa al database `foo`

Per lavorare su una collezione:

- `db.[collectionName].[method]([parameters])`

# Comandi principali

---

## Interrogazione dei dati

- Find, FindOne – modalità semplici per effettuare letture con proiezioni e selezioni
- Count, Distinct – modalità semplici per effettuare aggregazioni di dati
- Aggregate – modalità avanzata per effettuare aggregazioni di dati attraverso la concatenazioni di operazioni più semplici (match, unfold, group, ecc.)

## Modifica dei dati

- Insert, Delete, Update