Introduzione a MongoDB

MongoDB è un database NoSQL ampiamente utilizzato, specialmente per la gestione di dati semi-strutturati e per scenari in cui la flessibilità della struttura dei dati è essenziale.

8.1 - I NoSQL Databases

Il modello relazionale e i DBMS basati su SQL hanno funzionato egregiamente per anni, ma l'avvento dei Big Data e dei formati semi-strutturati come XML e JSON ha evidenziato i loro limiti, in particolare la rigidità nello schema dei dati.

I database NoSQL, e in particolare i JSON Databases come MongoDB, offrono un'alternativa più flessibile e scalabile.

NoSQL sta per "Not Only SQL", ovvero il fatto che non esiste solo SQL ma anche altro; I cosiddetti JSON Databases, o JSON Stores, sono una particolare categoria dei database NoSQL.

8.2 - MongoDB

MongoDB e' sicuramente il database NoSQL piu famoso ed utilizzato, è disponibile in due versioni:

- *DBMS da installare*: scaricabile dalla pagina ufficiale
- *Servizio cloud Atlas*: disponibile su MongoDB Atlas

MongoDB utilizza un modello di dati basato su documenti JSON. Le principali entità sono:

- *Database*: insieme di collezioni.
- *Collezione*: insieme di documenti JSON.
- *Documento*: unità base di dati, con struttura flessibile e senza schema fisso.
- *Identificatore univoco (_id)*: ogni documento ha un campo `_id` che lo identifica univocamente.

Strumenti per la gestione di MongoDB

MongoDB può essere gestito tramite:

- *CLI "mongo"*: (solo per Linux).
- *Tool grafici*: come Robo 3T (ex RoboMongo), scaricabile da robomongo.org.
- *Studio 3T*: un tool più avanzato.

Linguaggio di Query per MongoDB: MQL

MongoDB Query Language (MQL) differisce da SQL e si basa su JSON. Le query utilizzano una sintassi a oggetti simile a JavaScript. Viene utilizzato un oggetto javascript di base `db` che fa riferimento all'intera base di dati (tutte le collezioni).

- **Accesso alle collezioni**:
  - Ogni collezione e' un campo dell'oggetto `db` con nome differente dalle altre collezioni `db.MyCollection`
  - in alternativa `db.getCollection('MyCollection')`
- **Inserimento dati**:
  - `db.MyCollection.insertOne({name: "Pippo"})` Inserisce l'oggetto nella collezione. Restituisce un oggetto che riporta l'ID del documento creato.
  - `db.inventory.insertMany([{item: "journal", qty: 25}, {item: "notebook", qty: 50}])` Inserisce un array di oggetti nella collezione. Restituisce un oggetto con l'elenco degli ID creati.
- **Ricerca dati**:
  Il piu utilizzato e' il `find` , l'argomento è un documento che specifica le condizioni di selezione.
  - `db.inventory.find({})` (restituisce tutti i documenti)
  - `db.inventory.find({ status: "D" })` (filtra per un valore specifico)
    Si puo inoltre utilizzare una sintassi del tipo `{ <field1>: { <operator1>: <value1> }, ... }` , dove l'operatore è un campo il cui nome inizia con `$` (come `$ne` per l'operatore `!=` e `$eq` per l'operatore `==`), per estrarre elementi con determinati parametri dal db.

Operatori di Query

MongoDB offre numerosi operatori per filtrare i dati:

- **Confronto**:
  - `$eq` , `$ne` , `$gt` , `$gte` , `$lt` , `$lte`
- **Liste**:
  - `$in` , `$nin`
- **Operatori logici**:
  - `$and` , `$or` , `$nor` , `$not`
- **Esistenza e tipi di dati**:
  - `$exists` , `$type`

Tutti gli operatori: https://docs.mongodb.com/manual/reference/operator/query/

Operazioni su array e documenti annidati

MongoDB permette di gestire documenti con array annidati:

- `$elemMatch` : verifica condizioni su elementi di un array.
- *Documenti annidati*:
  - Ricerca con match esatto:

    ```
    db.inventory.find({size: {h: 8.5, w: 11.0, uom: "in" }})
    ```

  - Ricerca con dot notation:

    ```
    db.inventory.find({ "size.h": 8.5, "size.uom" : "in" })
    ```

Alcune funzioni possono essere applicate in coda alle funzioni di `find` :

- `limit(n)` : limita a n il numero di oggetti nel risultato.
- `skip(n)` : salta i primi n elementi nel risultato.
- `count()` : conta gli elementi nel risultato
- `sort({<field>: <option>, …} )` : ordina gli oggetti del risultato in base alle chiavi di ordinamento specificate, il valore `<option>` può valere:
  - `1` (ordine ascendente)
  - `-1` (ordine discendente)

ATTENZIONE: `find` non restituisce oggetti di tipo `«collection»` . Invece, restituisce dei `«cursor»` , di conseguenza, i metodi che si possono usare dopo `find` non sono gli stessi che si possono usare sulle collezioni base.

Modifica dei dati e metodi delle collezioni

MongoDB permette di aggiornare e cancellare documenti con metodi specifici:

- **Update**:
  Aggiornano il contenuto dei documenti selezionati. Le espressioni possono essere complicate.
  - `db.inventory.updateOne({status: "D"}, {$set: { qty: 10}})`
  - `db.inventory.updateMany({qty: {$lte: 15}}, {$set: { refurnish: true}})`
- **Delete**:
  - `db.inventory.deleteMany({status: "D"})` : cancella i documenti che soddisfano la condizione.