
TINVENTION
MongoDB Expirience

MongoDB - Presentazione

MongoDB è un database di tipo non relazionale (noSql), precisamente di tipo documentale.

Il concetto di base quindi diventa il *documento*, che può essere paragonato grossomodo ad una riga di un database relazionale.

MongoDB - Presentazione

Ogni documento è contenuto in una *collection*, che può essere grossomodo paragonabile alle tabelle del modello relazionale.

Le collection sono poi i contenitori capaci di eseguire le operazioni sui propri documenti.

Per la mia poca esperienza non sono previste dagli sviluppatori di mongo operazioni inter-collection.

MongoDB - Caratteristiche

Caratteristiche principale:

•Query ad hoc: supporta ricerche per campi, intervalli anche con REGEX

•Indicizzazione: disponibilità anche di indici full text

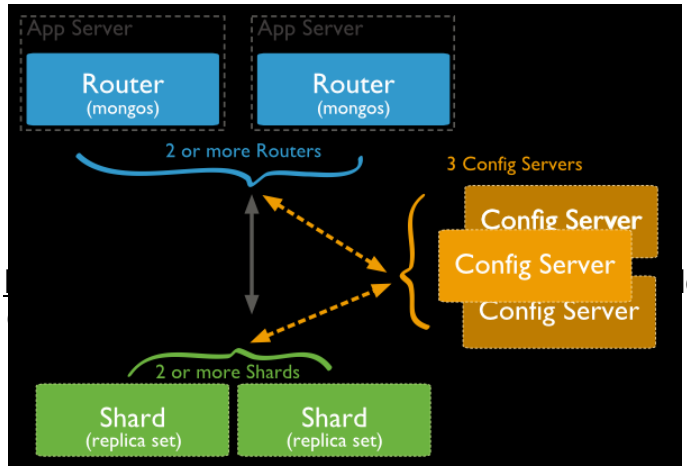
•Aggregazione dei dati: con map-reduce o aggregation framework.

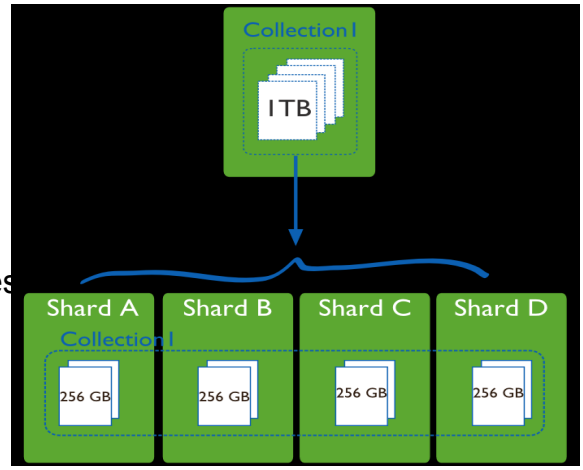
•Capped Collection: collection di dimensioni fisse.

•Alta Affidabilità: alta disponibilità grazie ai replica set, copie multiple dei dati gestite dal DBMS. Il dato può essere primario (su cui vengono eseguite le operazioni) o secondario (di Backup) con algoritmi automatici che ripristinano la copia secondaria in quella primaria in caso di errore.

MongoDB - Caratteristiche

- Sharding e bilanciamento dei dati: scala orizzontale dei dati e bilanciamento automatico tra gli shard. Utilizza delle chiavi di sharding per la gestione delle copie.





MongoDB - Caratteristiche

Ulteriori caratteristriche che non affronterò perche analaghe al modello relazionale sono:

•Security:
http://docs.mongodb.org/manual/core/security-introduction/

•Aggregation:
http://docs.mongodb.org/manual/core/aggregation-introduction/

•Indexes:
http://docs.mongodb.org/manual/core/indexes-introduction/

MongoDB - Documenti

In mongoDb il documento è sostanzialmente un documento Bson, che consiste in coppie chiave-valore dove la chiave(campo) è univoca nel documento e il valore è un qualunque valore (stringhe, interi , date…), compresi array e documenti.

```
{
  name: "sue",                    ← field: value
  age: 26,                        ← field: value
  status: "A",                    ← field: value
  groups: [ "news", "sports" ]    ← field: value
}
```

MongoDB - Documenti

E’ possibile infatti usare come valore un sottodocumento.

Esiste un campo (chiave) speciale aggiunto in automatico in ogni documento, chiamato _id di tipo ObjectId che corrisponde all’id univoco del documento.

Esempio:

```
{
_id: ObjectId("5099803df3f4948bd2f98391"),
name: { first: "Alan", last: "Turing" },
birth: new Date('Jun 23, 1912'),
death: new Date('Jun 07, 1954'),
contribs: [ "Turing machine", "Turing test", "Turingery" ],
views : NumberLong(1250000)
}
```

ObjectId come id

Sottodocumento con campi first e last

Array

MongoDB - Documenti

La cosa più innovativa di mongo è l'assenza di una struttura prestabilità della collection, ma ogni documento può avere tutti i campi diversi da un altro documento della stessa collection.

La struttura del documento non è quindi decisa a priori ne univoca, ma viene definita quando si aggiunge un documento.

MongoDB - Read

In mongoDb non esiste il concetto di SQL, ma le consultazioni e le modifiche vanno fatte con degli esempi e lanciate con comandi ad hoc, un esempio di query:

```
db.users.find(                          <-- collection
   { age: { $gt: 18 } },                <-- query criteria
   { name: 1, address: 1 }              <-- projection
).limit(5)                              <-- cursor modifier
```

La stessa operazione in SQL:

```
SELECT _id, name, address   <-- projection
FROM   users                <-- table
WHERE  age > 18             <-- select criteria
LIMIT  5                    <-- cursor modifier
```

MongoDB - Query

Esistono gli operatori per indicare minore, maggiore ecc.
Questi operatori sono dai inserire nelle query come ad esempio:

```
db.inventory.find(
{
type: 'food',
price: { $lt: 9.95 }
} )
```

Ecco alcuni degli altri operatori disponibili:

$gt	Matches values that are greater than the value specified in the query.
$gte	Matches values that are greater than or equal to the value specified in the query.
$in	Matches any of the values that exist in an array specified in the query.
$lt	Matches values that are less than the value specified in the query.
$lte	Matches values that are less than or equal to the value specified in the query.
$ne	Matches all values that are not equal to the value specified in the query.
$nin	Matches values that **do not** exist in an array specified to the query.

MongoDB - Query

Naturalmente esistono le operazioni di AND e OR:

Condizione di AND:

```
db.inventory.find(
{
type: 'food',
price: { $lt: 9.95 }
} )
```

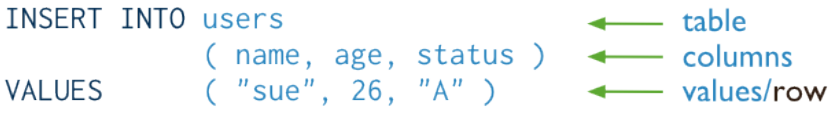
Condizione di OR:

```
{
$or: [ { qty: { $gt: 100 } }, { price: { $lt: 9.95 } } ]
}
```

MongoDB - Create

Per inserire un nuovo documento in una collection, ad esempio in *users,* dobbiamo lanciare il comando insert inserendo nelle parentesi il documento completo senza _id, da inserire.
Esempio:

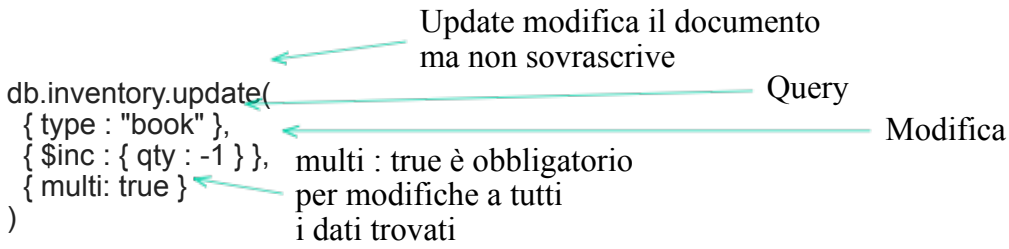
La stessa operazione in SQL:



MongoDB - Update

Esistono anche le operazioni di Update e Delete

Esempio di remove all e remove condizionato:



MongoDB - Delete

Esempio di remove all e remove condizionato:

```
db.inventory.remove({})
db.inventory.remove( { type : "food" } )
```