MongoDB

A DOCUMENT-ORIENTED DATABASE

Manipolazione dei dati

INSERT, UPDATE, DELETE

Inserire documenti raw

Inserire un documento

- `db.foo.insert({"bar":"baz"})`

Inserire più documenti

- `db.foo.batchInsert([{"_id":0}, {"_id":1}, {"_id":2}])`
- Se fallisce l'inserimento di un documento intermedio (e.g., il secondo), l'inserimento si interrompe (i.e., il terzo non viene inserito) ma i documenti precedente vengono memorizzati (i.e., il primo viene memorizzato)

I documenti non devono superare i 16MB di grandezza

Il campo `_id` viene inserito automaticamente se non indicato

Inserire documenti da file

Inserire i documenti presenti in un file

- `mongoimport --db [nomeDb] --collection [nomeCol] --file [filePath.json] [--jsonArray]`
- Compatibile con l'output di `mongoexport`
- L'opzione `--jsonArray` va usata se il file è strutturato come un array di documenti

Ripristinare una collection

- `mongorestore --collection [nomeCol] --db [nomeDb] [directory]`
- Vengono automaticamente cercati i file con lo stesso nome della collection
- Compatibile con l'output di `mongodump`

Nota: `mongoimport`, `mongoexport`, `mongorestore` e `mongodump` sono tutte applicazioni da lanciare da linea di comando

Rimuovere documenti

Rimuovere tutti i documenti in una collezione

- `db.foo.remove()`

Rimuovere i documenti che rispondo a determinati criteri

- `db.mailing.list.remove({"opt-out" : true})`

Cancellare una collection

- `db.tester.drop()`
- Svuotare una collection cancellandola e ricreandola richiede molto meno tempo rispetto alla cancellazione di tutti i documenti

Attenzione: non esiste un rollback! Una cancellazione è per sempre…

Aggiornare documenti

L'aggiornamento di un documento avviene in maniera atomica

- Due aggiornamenti concorrenti sullo stesso documento vengono comunque serializzati; **l'ultimo che arriva, vince**

Esistono diversi modi diversi per aggiornare i documenti

- Sostituzione del documento con un altro documento
- Aggiornamento dei campi attraverso «modificatori»
- Opzione upsert
- Funziona save da shell

Aggiornare documenti

La forma di base è:

- db.collection.update(objSelection, objUpdate, [objOptions])

Dove:

- objSelection è l'oggetto con i criteri necessari per individuare il/i documento/i da aggiornare
- objUpdate è l'oggetto con le modifiche da apportare
- objOptions è un eventuale oggetto in cui specificare una serie di opzioni

Di default, l'update aggiorna solamente 1 documento (il primo che trova che corrisponde ai criteri indicati)

- Per aggiornare tutti i documenti che corrispondo ai criteri, è necessario specificare l'opzione multi: true all'interno di objOptions
- Dato che il comportamento di base può essere cambiato in futuro, è buona norma specificare multi: false piuttosto che non specificare nulla

Aggiornare documenti – Sostituzione

Il modo più semplice è quello di **rimpiazzare l'intero documento**

- > joe = db.people.findOne({"name" : "joe", "age" : 20});
- {
  "_id" : ObjectId("4b2b9f67a1f631733d917a7c"),
  "name" : "joe",
  "age" : 20
  }
- > joe.age++;
- > db.people.update({"name" : "joe"}, **joe**);

Attenzione: se lo stesso nome compare in più documenti, si rischia di aggiornare il documento sbagliato

- In questo caso, **è buona norma utilizzare l'_id** come criterio di ricerca

Aggiornare documenti – Sostituzione

E’ possibile anche utilizzare una forma più compatta

- db.people.update(
  {"name" : "joe", "age" : 20},
  {
  "_id" : ObjectId("4b2b9f67a1f631733d917a7c"),
  "name" : "joe",
  "age" : 21
  }
)

Attenzione: se non si indica lo stesso _id, ne viene generato uno nuovo

Aggiornare documenti – Modificatori

I modificatori sono chiavi speciali (simili agli operatori di ricerca) che **specificano il tipo di modificare da effettuare** sui documenti

I modificatori sono:

- $set, $unset – Imposta il valore di un campo
- $inc – Incrementa (o decrementa) il valore di un campo numerico
- $push, $addToSet – Aggiunge un valore ad un array
- $pop, $pull – Rimuove un valore da un array

Aggiornare documenti – Modificatori $set e $unset

$set imposta il valore di un campo; **se non esiste, lo crea**

Alcuni esempi:

- db.users.update({"_id" : ObjectId("4b253b067525f35f94b60a31")},
  {"$set" : {"favorite book" : "War and Peace"}})
- db.blog.posts.update({"author.name" : "joe"},
  {"$set" : {"author.name" : "joe schmoe"}})

$unset rimuove la coppia chiave-valore dal documento

- db.users.update({"name" : "joe"}, {"$unset" : {"favorite book" : 1}})

Attenzione: per aggiornare i singoli campi bisogna sempre usare i modificatori, altrimenti diventa un aggiornamento per sostituzione!

- db.coll.update(*criteria*, {"foo" : "bar"})

Aggiornare documenti – Modificatore $inc

$inc aumenta il valore di un **campo numerico** della cifra indicata

- Se si indica una cifra negativa, il valore viene decrementato
- Se si applica su un campo non numerico, la query restituirà un errore
- **Se il campo non esiste, viene creato assegnando come valore la cifra indicata**
- Se non si indica una cifra (e.g., si indica una stringa o una data), la query restituirà un errore

Esempio:

- db.games.update({"game" : "pinball", "user" : "joe"}, {"$inc" : {"score" : 50}})

Aggiornare documenti – Modificatore $push

$push aggiunge valori in fondo ad un array

- Se il campo non esiste, viene creato
- **Se il campo esiste ma non è un array, viene prima convertito in un array e poi il campo viene aggiunto**
- E' possibile aggiungere più elementi con un unico comando utilizzando anche l'operatore $each

Esempi:

- db.blog.posts.update({"title" : "A blog post"},
  {"$push" : {"comments" :
  {"name": "bob", "email": "bob@ex.com", "content": "good post."}
  } })
- db.stock.ticker.update({"_id" : "GOOG"},
  {"$push" : {"hourly" : {"$each" : [562.776, 562.790, 559.123]} } })

Aggiornare documenti – Modificatore $addToSet

$addToSet funziona **come il $push, ma** con una differenza: i valori vengono aggiunti all'array **solo se non sono già presenti** al suo interno

Esempi:

- db.users.update({"_id" : ObjectId("4b2d75476cc613d5ee930164")},
  {"$addToSet" : {"emails" : "joe@gmail.com"}})
- db.users.update({"_id" : ObjectId("4b2d75476cc613d5ee930164")},
  {"$addToSet" : {"emails" : {"$each" :
  ["joe@php.net", "joe@example.com", "joe@python.org"] } } })

In alternativa, è possibile utilizzare $push in combinazione con $ne per verificare se il valore da aggiungere esisteva già o meno

- db.papers.update({"authors cited" : {"$ne" : "Richie"}},
  {$push : {"authors cited" : "Richie"}})

Aggiornare documenti – Modificatori $pop e $pull

$pop permette di **estrarre il primo o l'ultimo valore** nell'array (come se fosse una coda o una pila)

$pull permette di **estrarre uno o più valori** in base a determinati criteri

Esempi:

- db.lists.update({}, {"$pop" : {"todo": 1} })
  Estrae l'ultimo elemento nell'array todo
- db.lists.update({}, {"$pop" : {"todo": -1} })
  Estrae il primo elemento nell'array todo
- db.lists.update({}, {"$pull" : {"todo" : "laundry"}})
  Estrae tutti gli elementi nell'array todo che hanno valore "laundry"

Aggiornare documenti – Modificatori su array

Per modificare gli elementi di un array è possibile utilizzare i modificatori $set, $unset ed $inc insieme ad un indice posizionale

- Se si conosce la posizione dell'elemento da modificare, si può utilizzare direttamente quella
- **Se non si conosce la posizione dell'elemento da modificare, bisogna utilizzare l'operatore posizionale $**
- Gli array in MongoDB sono 0-based

Esempi:

- db.blog.update({"post" : post_id}, {"$inc" : {"comments.0.votes" : 1} })
- db.blog.update({"comments.author" : "John"},
  {"$set" : {"comments.$.author" : "Jim"} })
  Imposta Jim come autore del commento il cui autore è John
  Se esistono più commenti il cui autore è John, solo il primo viene modificato

Aggiornare documenti – Upsert

Upsert è un'opzione del comando update con la seguente semantica

- Se il criterio di ricerca trova un documento, aggiornarlo in base alle indicazioni date
- Altrimenti, crea un documento inizializzando i campi con i valori indicati

Esempio:

- db.analytics.update({"url" : "/blog"},
  {"$inc" : {"pageviews" : 1}}, { upsert: true })
  Se esiste un documento con url uguale a "/blog", aumenta di 1 le sue pageviews; altrimenti, crea un documento con una pageview

Se il valore di inizializzazione deve essere diverso da quello di aggiornamento, è possibile utilizzare l'operatore **$setOnInsert**

- db.users.update({}, {"$setOnInsert" : {"createdAt" : new Date()}},
  { upsert: true })

Aggiornare documenti – Funzione save

Se si sta utilizzando la **shell** di MongoDB, è possibile ricorrere al comando **save** per inserire o aggiornare un documento

- La funzione prende come unico parametro un documento
- **Se la collezione contiene già un altro documento con lo stesso _id, questo viene rimpiazzato con quello passato come parametro**
- Se invece non esiste, il documento viene aggiunto

Esempio

- > var x = db.foo.findOne()
- > x.num = 42
- > db.foo.save(x)

Senza save si può usare db.foo.update({"_id" : x._id}, x)