Tecnologie e applicazioni web

MongoDB

Filippo Bergamasco (filippo.bergamasco@unive.it)
http://www.dais.unive.it/~bergamasco/
DAIS - Università Ca'Foscari di Venezia
Anno accademico: 2017/2018

Cos'è MongoDB

MongoDB è un DBMS:

- Non relazionale
- Orientato ai documenti (non alle relazioni fra i dati)
- Con schema dinamico (schema-less)
- Documenti in stile JSON

Perché è interessante?
Si integra molto bene con linguaggi dinamici come javascript e non richiede una progettazione a-priori del database

Cos'è MongoDB

Database Relazionale

First	Last	Email	Twitter
Guillermo	Rauch	rauchg@gmail.com	rauchg

MongoDB:

```
{
     "name": "Guillermo"
   , "last": "Rauch"
   , "email": "rauchg@gmail.com"
   , "age": 21
   , "twitter": "rauchg"
}
```

```
   , "email": "rauchg@gmail.com"
   , "age": 21
   , "social_networks": {
         "twitter": "rauchg"
       , "facebook": "rauchg@gmail.com"
       , "linkedin": 27760647
     }
}
```

Caratteristiche

Query ad Hoc:

Supporto query per campi, intervalli ed espressioni regolari

Indicizzazione:

Qualsiasi campo dei documenti può essere indicizzato per velocizzare le query

Aggregazione:

Supporto per tipologie efficienti di aggregazioni sui dati (per calcolo di statistiche)

Caratteristiche

File storage:

Può essere utilizzato come un file-system distribuito (I files sono divisi in piccoli chunks e distribuiti su più nodi) GridFS

Sharding:

I dati di una collection possono essere distribuiti tra vari nodi di un infrastruttura Cloud sulla base di una chiave di sharding. Supporta inoltre meccanismi automatici di bilanciamento dei dati.

MongoDB

https://docs.mongodb.com/manual/#

Il DBMS permette la creazione di molteplici **database**.

Ciascun **database** è composto da più **collections**.
Ciascuna **collection** è composta da più **documents**
Ciascun **document** è composto da molti **fields**

MongoDB


Collections
{
"_id" : ObjectId("527b3cc65ceafed9b2254a94"),
"f_name" : "Zenny",
"sex" : "Female",
"class" :
"age" : 1
"grd_poi
}
Document1
{
"_id" : ObjectId("527b3cc65ceafed9b2254a95"),
"f_name" : "Paul",
"sex" : "Male",
"class" : "VII",
"age" : 13,
Document2
{
"_id" : ObjectId("527b3cc65ceafed9b2254a97"),
"f_name" : "Lassy",
"sex" : "Female",
"class" : "VIII",
"age" : 13,
"grd_point" : 28.2514
}
Document3

Terminologia

RDBMS		MongoDB
Database	→	Database
Table	→	Collection
Index	→	Index
Row	→	Document
Column	→	Field
Join	→	Embedding & Linking

MongoDB vs. relazionale

- Ciascun documento di una collezione non necessariamente deve essere composto dagli stessi campi (fields)
  - Questo fornisce estrema flessibilità nell'uso dei dati che possono essere memorizzati/caricati senza uno schema specifico
- Clascun documento può contenere altri documenti (Embedding)
  - Il meccanismo può essere sfruttato al posto delle normali operazioni di Join tra tabelle relazionali

Relazioni one-to-many

Due modi per realizzarle:

1. Embeddando i documenti su una stessa struttura (Letture richiedono 1 query ma scritture lente)

2. Referenziando gli id dei documenti come nei database relazionali (Letture richiedono 2 query ma scritture più veloci)

Relazioni one-to-many

```
> book = db.books.find({ _id : "123" })
{
    _id: "123",
    title: "MongoDB: The Definitive Guide",
    authors: [
        { first: "Kristina", last: "Chodorow" },
        { first: "Mike", last: "Dirolf" }
    ],
    published_date: ISODate("2010-09-24"),
    pages: 216,
    language: "English",
    publisher: {
        name: "O'Reilly Media",
        founded: 1980,
        locations: ["CA", "NY" ]
    }
}
```

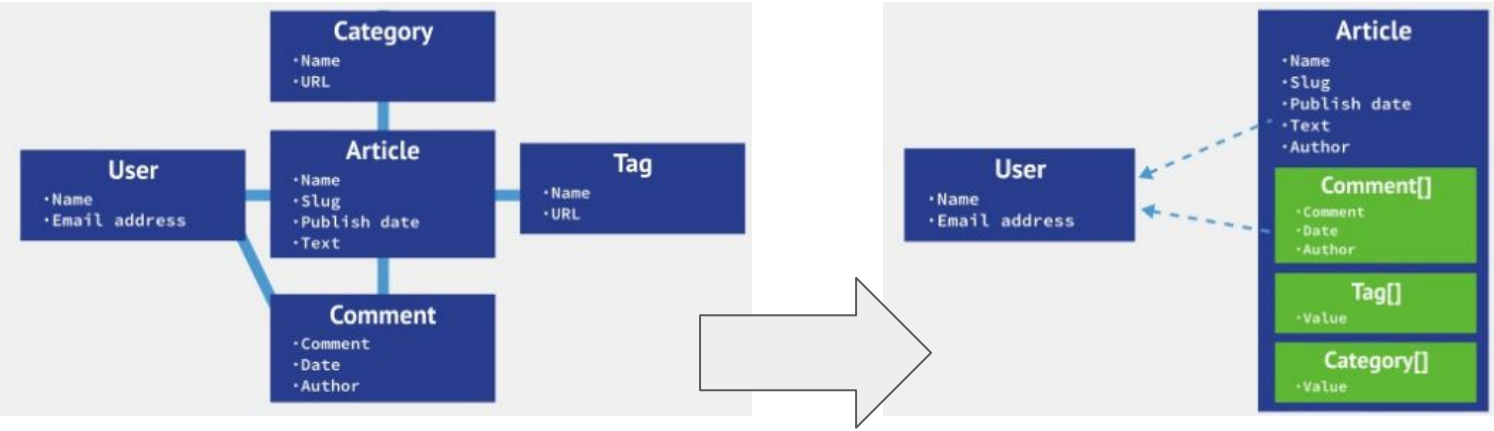
```
publisher = {
    _id: "oreilly",
    name: "O'Reilly Media",
    founded: "1980",
    location: "CA"
}

book = {
    title: "MongoDB: The Definitive Guide",
    authors: [ "Kristina Chodorow", "Mike Dirolf" ],
    published_date: ISODate("2010-09-24"),
    pages: 216,
    language: "English",
    publisher_id: "oreilly"
}
```

Relazioni one-to-many

Quale soluzione preferire? La scelta è dello sviluppatore che può decidere in base all'utilizzo tipico dell'applicazione

Esempio: piattaforma di blogging (molte letture, poche scritture)



MongoDB Shell

Insieme al DBMS viene fornito un semplice programma command-line per eseguire operazioni CRUD (Create, Read, Update, Delete)

Sintassi Javascript-like con API simili a quelle utilizzabili per nodejs.

```
$ mongo
> show log global;
```

MongoDB Shell

Comando	Descrizione
> show dbs	Visualizza la lista dei database
> use <db>	Cambia il database corrente
> show collections	Visualizza tutte le collezioni del database corrente
> db.<collection>.find()	Visualizza tutti i documenti di una collezione

Query

Operazioni di lettura sui documenti vengono effettuate fornendo dei **documenti** speciali chiamati Query Filter Documents

```
{
  <field1>: <value1>,
  <field2>: { <operator>: <value> },
  ...
}
```

Query

Esempi:

```
db.inventory.find({ status: "A", qty: { $lt: 30 }})
```

```
SELECT * FROM inventory WHERE status = "A" AND qty < 30
```

```
db.inventory.find({ status: "A" },{ item: 1,status: 1 })
```

```
SELECT _id, item, status from inventory WHERE status = "A"
```

Operazioni atomiche

Le operazioni in MongoDB sono **atomiche a livello di documento** (e di tutti i nested documents)

- Le transazioni che coinvolgono la modifica di più documenti vanno implementate manualmente con il pattern two-phase-commits

Un client può osservare le modifiche effettuate ad un documento prima che queste siano rese permanenti (read uncommitted)

Utilizzo in Node.js

E' possibile usare MongoDB in modo simile alla shell con il driver ufficiale per Node.js:

https://www.npmjs.com/package/mongodb

- Inserire "mongodb" come dipendenza in package.json
- Recuperare l'oggetto `MongoClient` per connettersi al database e lavorare con le collections

Mongoose

Mongoose è una libreria molto utilizzata di Object Document Mapping **ODM** tra Javascript e MongoDB

Permette di definire lo schema dei documenti attraverso oggetti Javascript per effettuare il mapping automatico da e verso il database

http://mongoosejs.com/docs/guide.html

Mongoose

Le unità di lavoro principali di Mongoose sono:

Schemas:

Descrivono la struttura dei documenti di una determinata collezione

Models:

Sono delle funzioni (costruttori) che permettono di costruire oggetti, dato un certo schema, e memorizzarli nella rispettiva collezione nel database

Mongoose models

Una volta definito lo schema e creato un certo modello è possibile utilizzarlo per:

- Effettuare query nel database
  - Ex: <model>.find({ })
- Creare un nuovo oggetto
  - Ex: <model>.create({obj})
- Rimuovere oggetti
  - Ex: <model>.remove({})