MongoDB

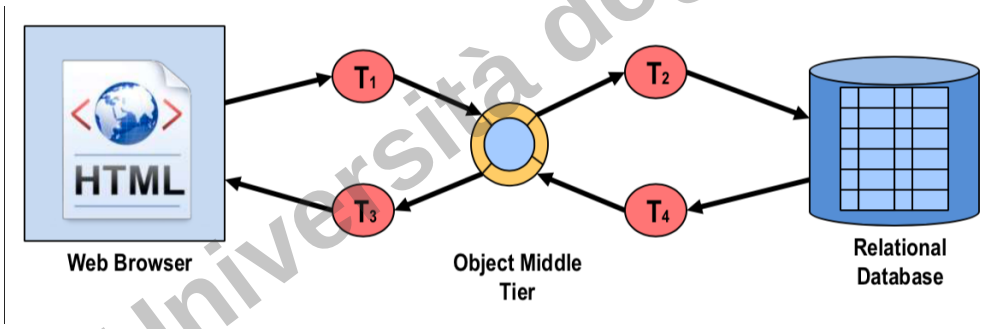
Università degli Studi di Milano

Programmazione Web e _Mobile

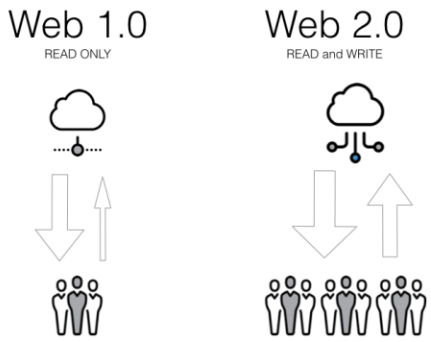
IL WEB 1.0 vs WEB 2.0

Nel Web 1.0 i dati consumati dalle applicazioni venivano prodotti a mano (HTML) oppure acquisiti da database relazionali attraverso la mediazione di un linguaggio di programmazione

Prima che i dati possano essere consumati da una applicazione Web servono quattro passaggi di trasformazione

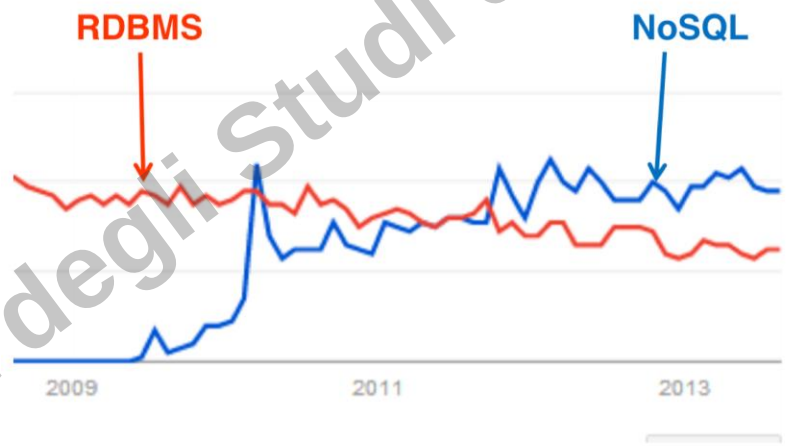


- Il **modello relazione** si diffonde negli anni 70 in un contesto nel quale le applicazioni sono centralizzate
- Per gestire i dati in contesti di applicazioni decentralizzate servono nuovi modelli



RDBMS vs NoSQL

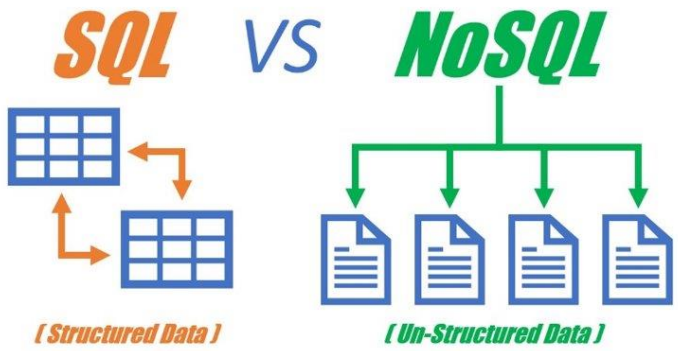
NoSQL: Non ha relazioni come SQL, non ha struttura, non ha le tabelle



http://www.google.com/trends/explore?q=NoSQL%2C+RDBMS#q=NoSQL%2C%20RDBMS&cmpt=q

tipi del NoSQL:

- Document databases.
- Key-value stores.
- Column-oriented databases.
- Graph databases.



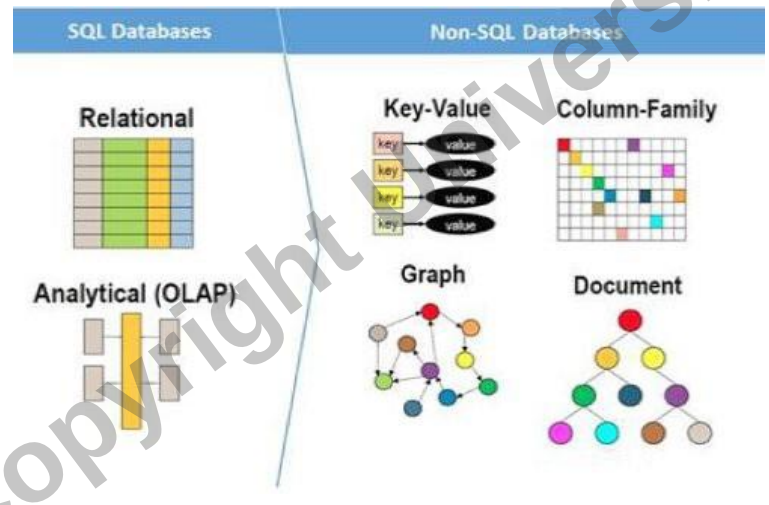
Cos'è MongoDB?

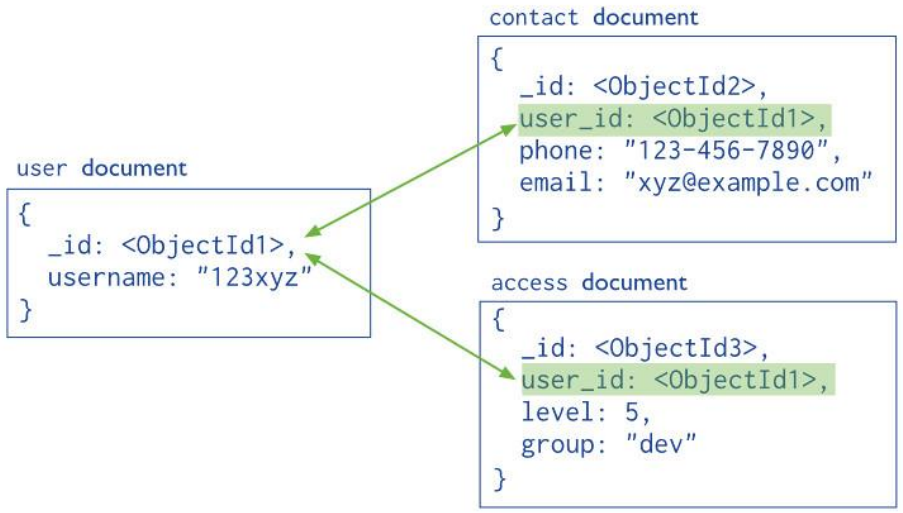
- Definizione
  - È un database open source scritto in C ++
  - Un database NoSQL
  - NoSQL Senza schema
  - Composta da collezione di documenti

Cos'è MongoDB?

- BSON invece di JSON, permette un po' più di dati rispetto a JSON in forma di campo e valore con possibilità di usare delle **chiavi esterne**

- Una collezione è un insieme di documenti di uno stesso tipo: utenti, prodotti, …
- Supporta l'elaborazione di query ad hoc
- è utilizzato come database back-end per organizzazioni come Facebook, IBM, ecc





JSON - BSON

JSON

- "JavaScript Object Notation"
- Facile per umane da write/read, per il computer da parse/generate
- gli oggetti possono essere nested
- costruita su
  - coppie campo/valore
  - Elenco ordinato di valori

http://json.org/

BSON

- "Binary JSON"
- Binary-encoded serialization of JSON-like docs
- permette "referencing"
- La struttura incorporata riduce la necessità di join
- Obiettivi:
  - Leggero
  - Attraversabile
  - Efficiente (decodifica e codifica)

http://bsonspec.org/

Esempio di BSON:

```
{
"_id" :          "37010"
"city" :         "ADAMS",
"pop" :          2660,
"state" :        "TN",
"councilman" : {
                 name: "John Smith"
                 address: "13 Scenic Way"
                                    }
}
```

- Value serves as primary key for collection (chiave primaria per la collezione)
- Value is unique, immutable, and may be any non-array type.
- Default data type is ObjectId, which is "small, likely unique, fast to generate, and ordered." Sorting on an ObjectId value is roughly equivalent to sorting on creation time

MongoDB vs. SQL

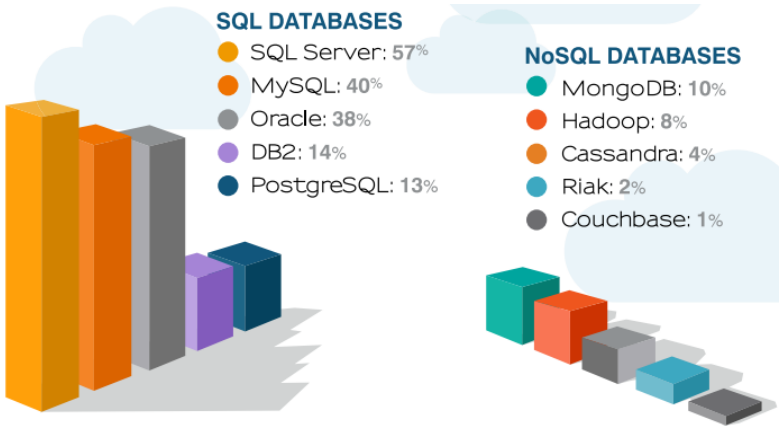
MongoDB	SQL
Document	Tuple
Collection	Table/View
PK: _id Field	PK: Any Attribute(s)
Uniformity not Required	Uniform Relation Schema
Index	Index
Embedded Structure	Joins
Shard	Partition

*PK: Primary Key

Perchè MongoDB

- È facile da installare, utilizzare, è un database senza schema
- Grazie alla capacità di un database senza schema, il codice che creiamo definisce lo schema
- Può archiviare qualsiasi tipo di dati con qualsiasi dimensione.
- I dati sono archiviati nel formato binary JSON che è una coppia chiave-valore, non è necessaria alcuna complessità di join.
- Utilizza la RAM per archiviare i dati,

questo consente un accesso più rapido ai dati



- Può essere replicato su più host e utilizzato
- Supporta la proprietà ACID (atomicità, coerenza, isolamento e durata) per una transazione del database.

Perchè MongoDB

- Supporta la replica, se il server primario si arresta durante la transazione, il server secondario gestisce la transazione senza interazione umana.
- È conveniente perché riduce i costi su hardware e archiviazione.
- Può salvare molti dati che aiuteranno ad elaborare le query più velocemente.
- Grazie allo schema dinamico, potete provare nuove cose a un costo inferiore. Non dovete preoccuparvi di preparare i dati prima di sperimentarli.

Perchè MongoDB

- Funziona perfettamente con ambienti di Big Data perché: ha la capacita di salvare grandi quantità di dati (volume), elaborandoli in modo veloce (velocità) senza la necessità di disporre di dati omogenei (varietà).
- La interfaccia facile con linguaggio commune (Java, Javascript, PHP, etc.)
- Puo usarla su VM, cloud,..
- Mantiene le caratteristiche essenziali degli RDBMS mentre apprende dai sistemi NoSQL a valore-chiave

MongoDB Data Model

- Document-Based (max 16 MB)
  - La dimensione massima del documento aiuta a garantire che un singolo documento non possa utilizzare una quantità eccessiva di RAM o, durante la trasmissione, una quantità eccessiva di larghezza di banda
- i documenti sono nel farmato di BSON, field-value pairs
- Ogni documento archiviato in una collezione
- Collections
  - hanno indice impostato in comune
  - Come tabelle relazionali dei db's.
  - I documenti non devono avere una struttura uniforme

-docs.mongodb.org/manual/

MongoDB Data Model

I documenti in MongoDB hanno le seguenti restrizioni sui nomi dei campi (field names):

- Il campo _id è riservato per l'uso come chiave primaria: il suo valore deve essere univoco nella collezione e può essere di qualsiasi tipo diverso da un array
- I nomi dei campi non possono iniziare con il carattere $
- I nomi dei campi non possono contenere Carattere .

CRUD:

Create, Read, Update, Delete

CRUD: Inserire dati nel nostro database usando shell

- Su shell con questi commandi:
  - **Show** dbs/collections
  - **Use** : usa il db se non esiste ve lo crea
  - **db.<Nome db>**: creare un db
  - db.<collection/db>.insert(<document/instance>)

    db.Nome.insertOne({“nome”:”Luca”, “cognome”:”Rossi”, “eta”:25})
  - db.Nome.insertMany([{},{}])

db.<collection>.insert(<document>) <=> INSERT INTO <table> VALUES(<attributevalues>);

Selezionare e filtrare database, operatori logici

- db.Nome.find()
  - Restituisce un cursore, che viene ripetuto sulla shell per visualizzare i primi 20 risultati.
  - .limit(<number>) per limitare i resultati

db.<collection>.find <=> SELECT * FROM <table>;

- db.Nome.find().pretty()
- db.Nome.findOne({citta:"Milano"}): il primo che trove me ne mande
- db.Nome.find({citta:"Milano"})
- Condizioni : Comparison operator $eq $gte $lt , $in, …
- Esempio: db.Nome.find({eta: {$gt:20}})
- db.Nome.find({eta:{$in:[21,25]}}).pretty()
- Operatori logici: $and, $not, $nor, $or
- Esempio: db.Nome.find({$and:[{eta:{$gt:22}},{citta:"Milano"}]})

```
db.<collection>.find({<field>:<value>})
"AND"
db.<collection>.find({<field1>:<value1>,

<field2>:<value2>
                                        })
```

```
SELECT *
FROM <table>
WHERE <field1> = <value1> AND <field2> =
<value2>;
```

BSON Types

Type	Number
Double	1
String	2
Object	3
Array	4
Binary data	5
Object id	7
Boolean	8
Date	9
Null	10
Regular Expression	11
JavaScript	13
Symbol	14
JavaScript (with scope)	15
32-bit integer	16
Timestamp	17
64-bit integer	18
Min key	255
Max key	127

I numeri possono essere usati con operatore di $type per query con il tipo

Modificare e aggiornare dati (update)

- db.Nome.update({nome:"Luca"},{$set:{eta: 29, cognome: "Rosso"}})
- db.Nome.update({_id:ObjectId("12357")},{$set: {eta: 29, cognome: "Rosso"}})
- db.Nome.updateMany({citta: "Milano"}, {$set: {cap:"45675"}})
- db.Nome.updateMany({}, {$inc: {eta:1}})
- Se un filtro non esiste fai nessuno, se esiste la modifica
- db.Nome.updateMany({nome:"Massimo"}, {$set:{cognome:"blu", eta: 23}},{upsert:true})

To remove a field

db.<collection>.update({<field>:<value>}, { $unset: { <field>: 1 }})

Replace all field-value pairs

db.<collection>.update({<field>:<value>},{ <field>:<value>, <field>:<value>})

Modificare e aggiornare dati (update)

```
db.<collection>.update(
{<field1>:<value1>},        //all docs in which field = value
{$set: {<field2>:<value2>}},                    //set field to value
{multi:true} )                    //update multiple docs
```

upsert: if true, creates a new doc when none matches search criteria.

```
UPDATE <table>
SET <field2> = <value2>
WHERE <field1> = <value1>;
```

Eliminare i dati

- db.Nome.deleteOne({nome:”Massimo”})
- db.Nome.deleteOne({$and: [{dta:{$gt:20}}, {cita:”Roma”}]})
- db.Nome.deleteMany({citta:”Milano”})

db.<collection>.remove({<field>:<value>})

```
DELETE FROM <table>
        WHERE <field> =
<value>;
```

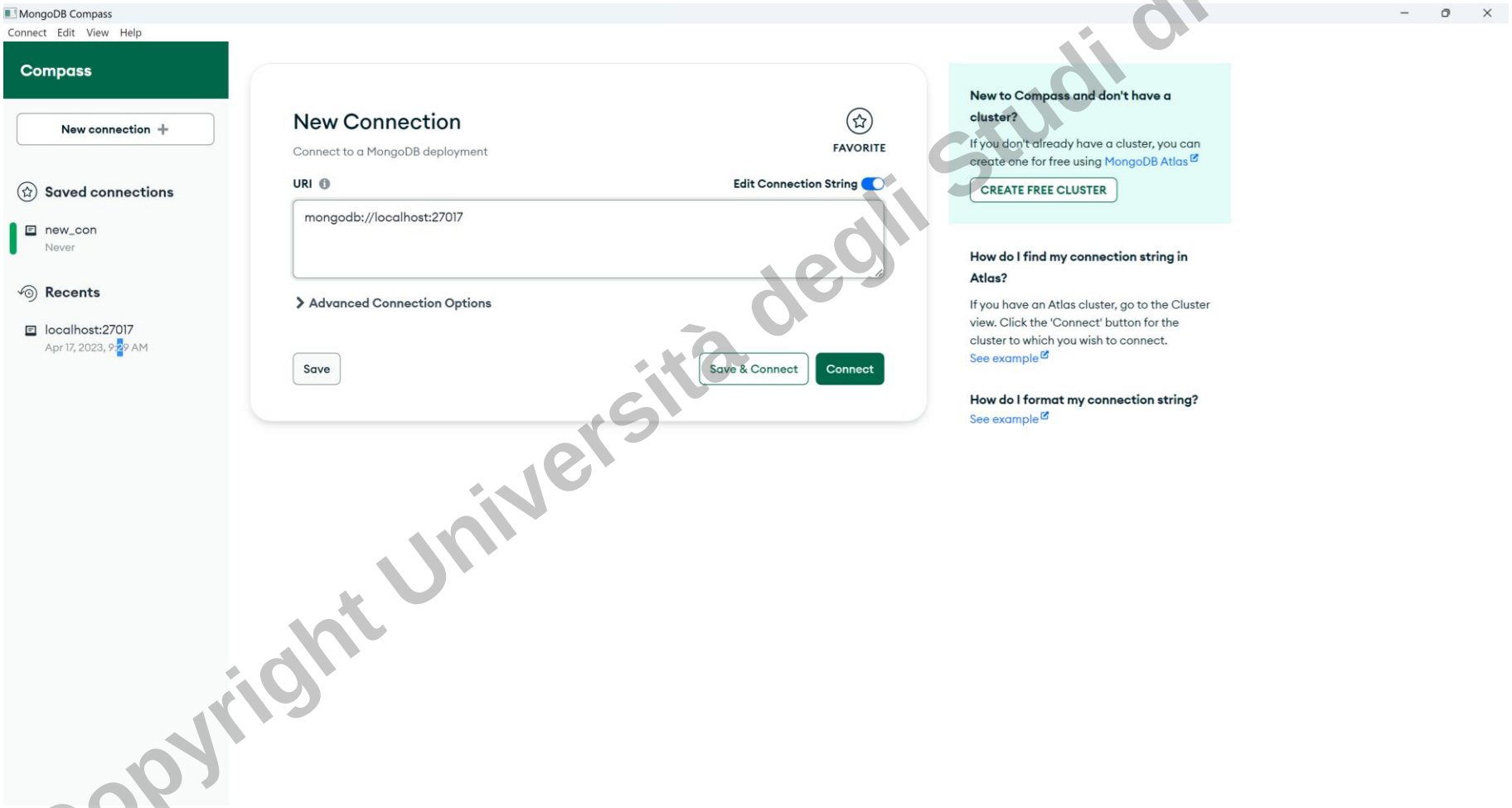
Isolation

- Per impostazione predefinita, tutte le scritture sono atomiche solo a livello di un singolo documento.
- Ciò significa che tutte le scritture possono essere intercalate con altre operazioni.
- Puoi isolare le scritture su una collezione unsharded aggiungendo **$isolated:1** nell'area delle query

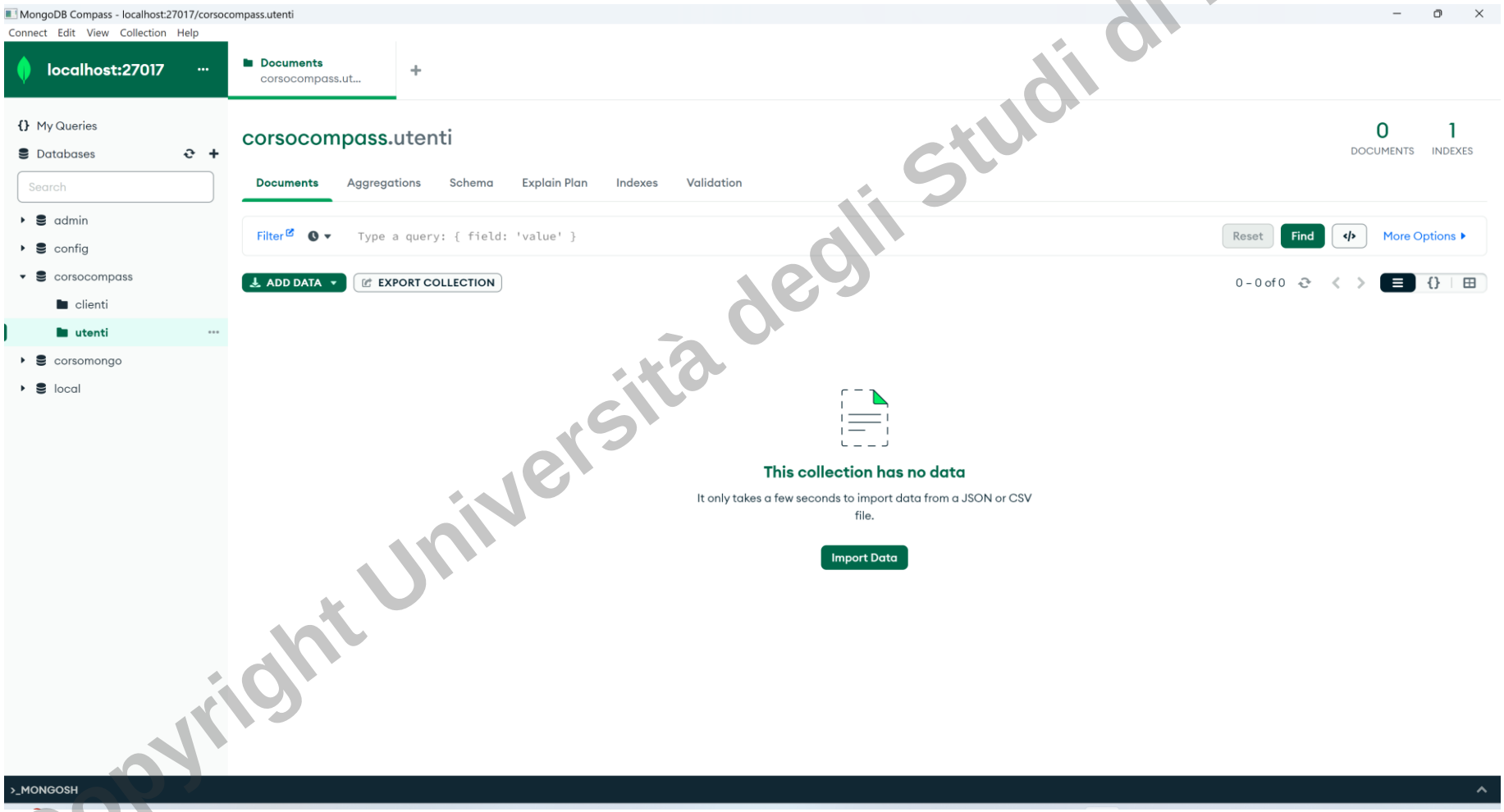
```
db.<collection>.remove({<field>:<value>,$isolated: 1})
```

MongoDB Compass

MongoDB Compass



MongoDB Compass



Conclusioni

- MongoDB è un database **affidabile** che è raccomandato durante la progettazione di un'applicazione Web che è **scalabile** e ha richiesto un database di grandi dimensioni per archiviare enormi quantità di dati non strutturati. Se l'utente è alla ricerca della migliore disponibilità, elaborazione più rapida, **buon backup**, zero perdita di informazioni, MongoDB è la soluzione migliore da utilizzare. soprattutto è gratis da usare.