Basi di Dati
Oltre i database relazionali

Slide rielaborate dalle slide di Ashwani Kumar, Livio Bioglio, Rosa Meo

Database non relazionali

- Cosa c'è oltre i database relazionali?
- Negli ultimi anni la disponibilità di enormi quantità di dati e la loro struttura non regolare ha fatto nascere approcci alternative al modello relazionale.
- Vediamo una brevissima panoramica.

Database non relazionali

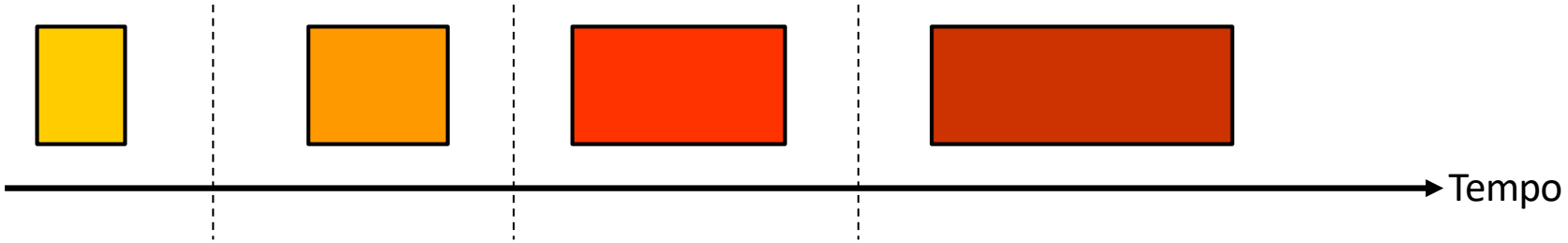
- NoSQL
  - Not only SQL (ma più propriamente bisognerebbe chiamarli “database non relazionali”)
- La maggior parte dei sistemi NoSQL sono database distribuiti
  - Attenzione su dati semi-strutturati, alte prestazioni, disponibilità, replicazione e scalabilità

Database non relazionali

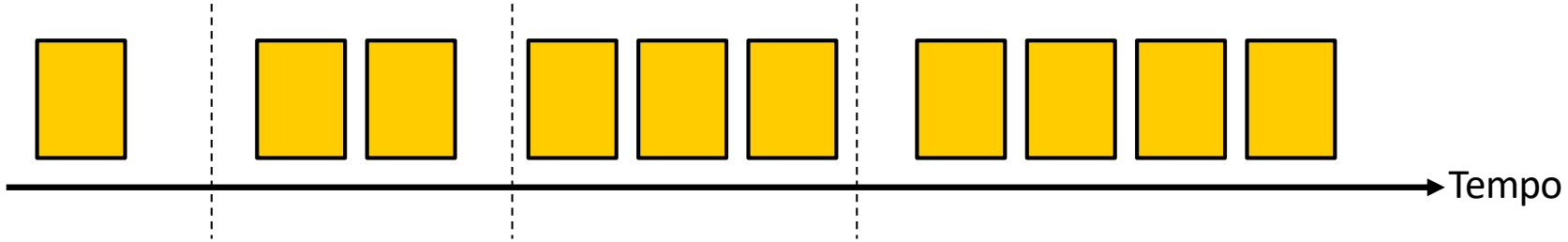
- NoSQL viene usato per i “big data”
- Applicazioni tipiche per NoSQL:
  - Social media
  - Web links
  - Profili utente
  - Marketing e vendite
  - Post e tweet
  - Mappe stradali e dati spaziali
  - Email

Scalabilità

Scalabilità verticale: aumentare la potenza di calcolo di una singola macchina migliorando le varie componenti (CPU, RAM, storage, ...).

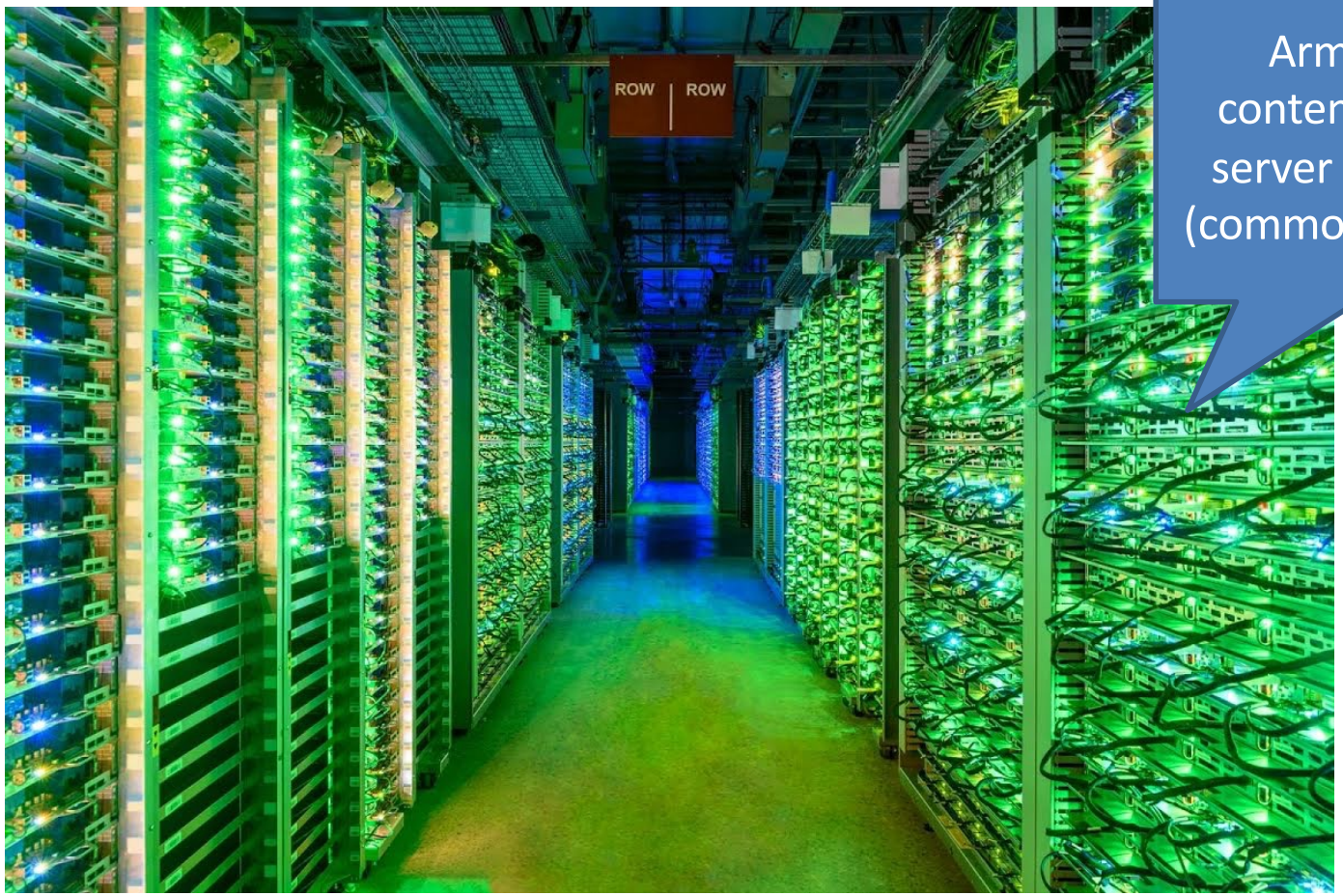


Scalabilità orizzontale: aumentare la potenza di calcolo aggiungendo nuove macchine non necessariamente più potenti di quelle già presenti.



- I database relazionali hanno una limitata scalabilità orizzontale cioè è difficile migliorare le loro performance aggiungendo nuove macchine al sistema.

Commodity machines



Armadio rack contenente diversi server poco costosi (commodity machine)

Data center di Google a St. Ghislain, in Belgio (https://www.google.com/about/datacenters/)

Caratteristiche degli approcci NoSQL

NoSQL evita:

- Costo delle proprietà ACIDE *(e non dà i vantaggi delle proprietà ACIDE)*
- Complessità delle query SQL *(e non dà la potenza delle query SQL)*
- Onere della progettazione a priori dello schema *(e non dà la garanzia della correttezza dei dati)*
- Transazioni *(ma devono essere gestite dagli sviluppatori a livello applicativo)*

NoSQL permette:

- Cambiamenti facili e frequenti al DB
- Sviluppo veloce
- Gestione di grandi quantità di dati
- Database senza schema

NoSQL: quando usarlo e quando evitarlo

NoSQL è utile quando

- Serve un sistema distribuito e scalabile
- Il modello relazionale è troppo restrittivo (serve uno schema flessibile o i dati hanno una struttura complessa)
- Le proprietà ACIDE non sono necessarie (in NoSQL gli aggiornamenti «prima o poi» sono eseguiti ma nel frattempo possono esserci letture inconsistenti)

Adatto, ad es., per:

- Log di dati da fonti distribuite
- Dati temporanei (carrelli, liste dei desideri, dati di sessione)

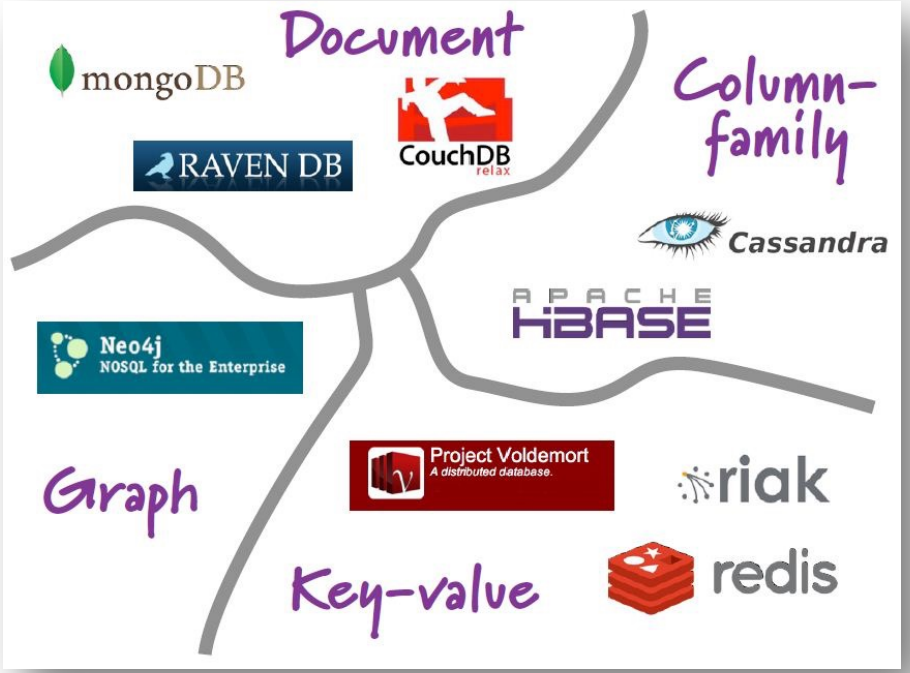
Non è adatto, per es., per:

- Dati finanziari
- Dati aziendali critici

Modelli dei dati

I database NoSQL hanno un panorama variegato. Si possono raggruppare secondo il modello di dati:

- **Chiave-valore**
- **Documento**
- **A colonna**
- **A grafo**



Non esiste una “lingua franca” come SQL: ogni database NoSQL ha il proprio linguaggio di query.

Quando usare NoSQL?

NoSQL non è un superamento dei DB relazionali ma un'alternativa utile in alcuni casi (eventualmente a supporto dei DB relazionali).

Vantaggi di NoSQL:

- **Alte prestazioni**, anche con enormi quantità di dati anche per l'assenza di join e transazioni;
- **Riduzione dei tempi di sviluppo**, dato che non hanno schema è sempre possibile aggiungere nuove strutture ai dati; inoltre la struttura dei dati può essere più vicina a quella usata nelle applicazioini (riduzione dell'impedance mismatch);
- Supporto alla **scalabilità orizzontale** e all'**alta disponibilità** grazie ai database distribuiti su server diversi.

Svantaggi di NoSQL:

- **Nessun supporto per join e transazioni**;
- **Grande ridondanza dei dati**;
- **Mancanza di un linguaggio di query standard**;
- **Mancanza di vincoli di integrità** (devono essere gestiti dagli sviluppatori a livello applicativo).

Caratteristiche dei database

Tipo di database	Caratteristiche	Esempi di DBMS
Relazionale	Query potenti e flessibili (SQL); I dati devono avere uno schema predefinito	Oracle, PostgreSQL, MySQL, SQL Server
Chiave-valore	Associa ogni chiave a un valore come fanno gli array associativi o tabelle hash dei linguaggi di programmazione; utili per fare cache in memoria principale per applicazioni web	Memcached (memcachedb, membase) Voldemort, Redis, Riak, Amazon Dynamo
Colonnario	Gestisce enormi volumi di dati su server diversi (nodi); Grande scalabilità orizzontale	Hbase, Cassandra, Hypertable, Google BigTable
Documentale	Un documento consiste in un ID associato a valori di vari tipi come ad esempio hash; può contenere strutture annidate	MongoDB, CouchDB
A grafo	I data sono altamente interconnessi e nodi e archi possono avere proprietà	Neo4J

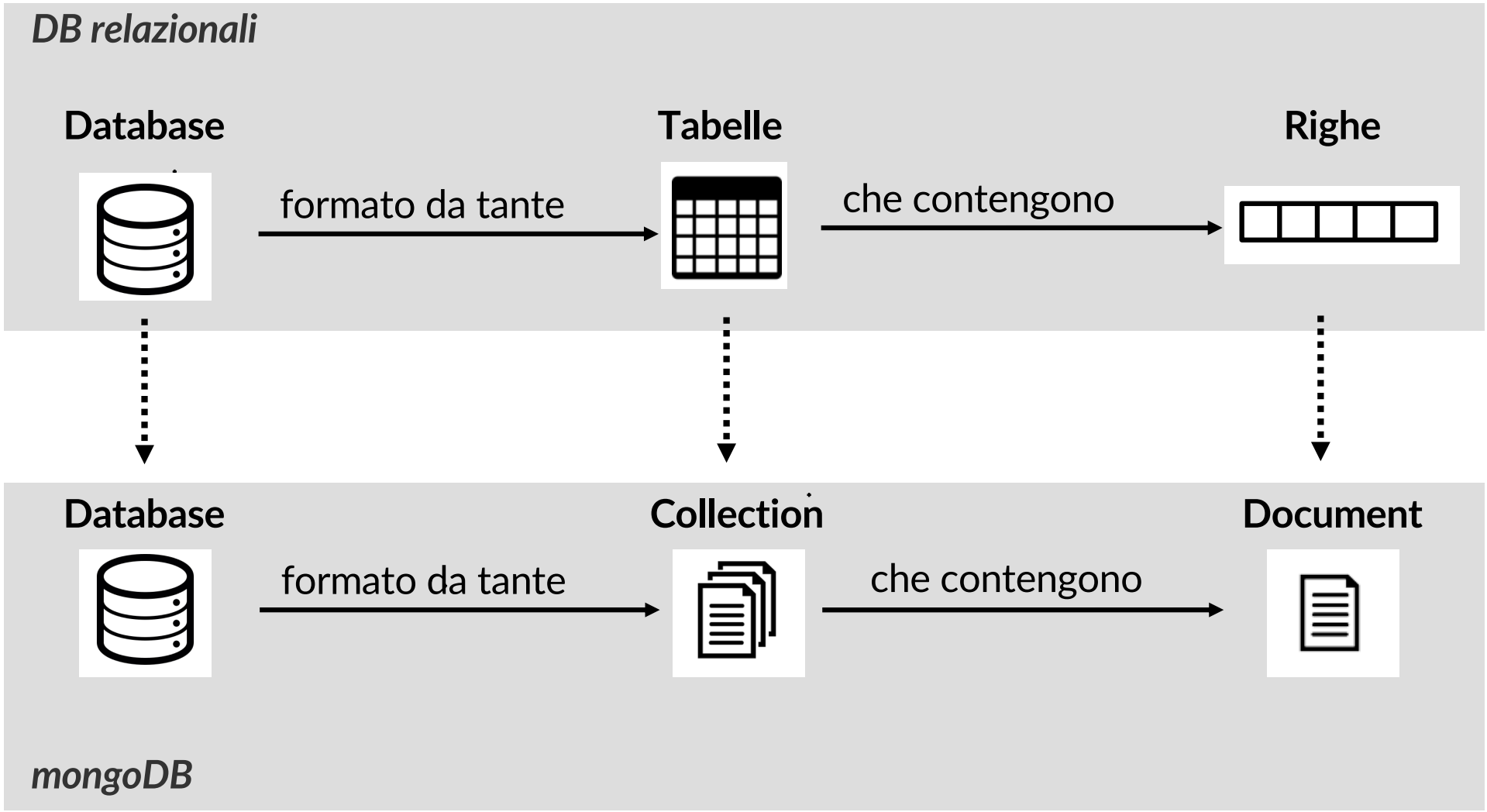
mongoDB tra i DB NoSQL

mongoDB è un database NoSQL orientato ai documenti. Il nome Mongo deriva da «humongous», «gigantesco».
Nato nel 2007, è diventato open source nel 2009. L'edizione Community Server è gratuita e disponibile per Windows, Linux, MacOS.
È uno dei DB NoSQL più diffusi.

In MongoDB:

- i dati sono memorizzati sotto forma di documenti, in formato JSON/BSON;
- i documenti non devono aderire ad uno schema standard, ma possono contenere qualsiasi campo;
- per le operazioni di ricerca, si recuperano i documenti basandosi sul valore di un determinato campo;
- il DB è scalabile orizzontalmente, supportando partizionamento (sharding) dei dati in sistemi distribuiti;
- esistono funzionalità per aggregazione e analisi dei dati.

Terminologia di mongoDB

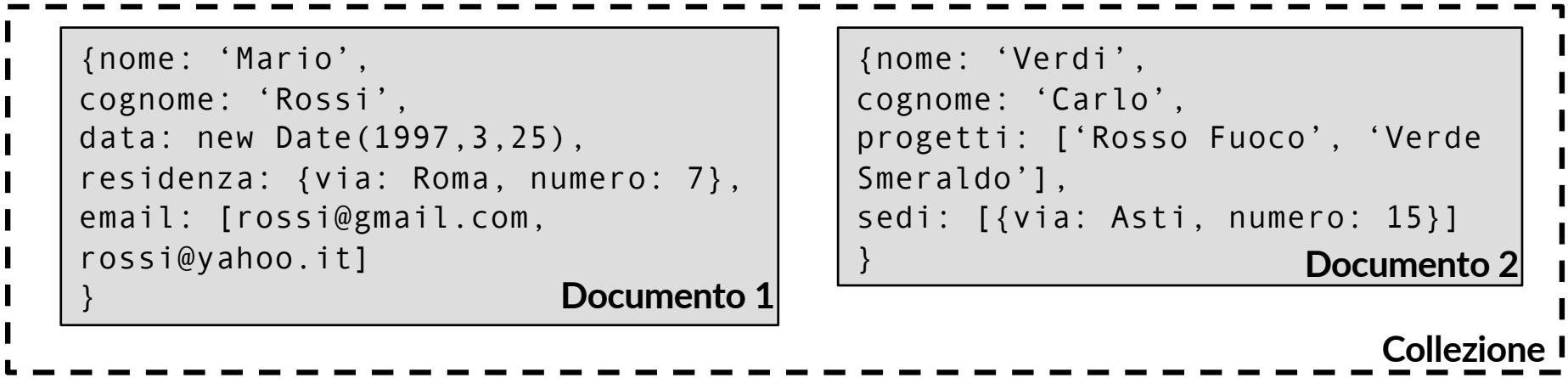


Documento

In mongoDB un documento è salvato in formato JSON (JavaScript Object Notation), un formato standard per lo scambio di dati tra applicazioni, soprattutto sul web.

- in realtà, il linguaggio JSON è usato per l'input/output delle query di aggiornamento o selezione, mentre per salvare i documenti internamente mongoDB usa il formato BSON, la codifica binaria di JSON.

I dati di un documento sono racchiusi tra {},
il singolo dato assume la forma *campo*: *valore*.



Lo schema non deve essere deciso a priori.
I documenti possono contenere anche campi diversi tra documenti della stessa collezione.
I valori possono essere tipi base (numeri, booleani, stringhe) ma anche array (racchiusi da []) e oggetti (altri JSON).

Esempio di query

Data una collezione sugli esami universitari:

```
{ "_id" : ObjectId("5d49b1d24847246e09ff1b18"), "matricola" : 0, "genere" : "F", "esame" : "prog1",
"voto" : 21, "anno" : 2013 }
{ "_id" : ObjectId("5d49b1d24847246e09ff1b19"), "matricola" : 0, "genere" : "F", "esame" : "db",
"voto" : 3, "anno" : 2006 }
{ "_id" : ObjectId("5d49b1d24847246e09ff1b1a"), "matricola" : 0, "genere" : "F", "esame" : "sisop",
"voto" : 6, "anno" : 2007 }
{ "_id" : ObjectId("5d49b1d24847246e09ff1b1c"), "matricola" : 1, "genere" : "F", "esame" : "db",
"voto" : 29, "anno" : 2014 }
…
```

Contiamo gli esami con voto 30 di ogni studente:

```
db.scores.aggregate([ {$match: {voto:30}},
 {$group: {_id:"matricola", count: {$sum: 1}}}])
```

La pipeline della funzione aggregate viene eseguita in ordine:
1) Seleziona solo (con *$match*) i documenti che hanno voto 30;
2) Raggruppa (con *$group*) i documenti dallo stage precedente per matricola (*_id:"matricola"*) sommando (*$sum*) il valore 1 per ogni documento nel gruppo.

Il comando è analogo alla query SQL del modello relazionale

select matricola, count(*)
from scores
where voto=30
group by matricola

Riferimenti

- https://www.mongodb.com/**nosql**-explained
- https://www.couchbase.com/**nosql**-resources/what-is-**no-sql**
- http://nosql-database.org/ "NoSQL DEFINITION: Next Generation Databases mostly addressing some of the points: being non-relational, distributed, open-source and horizontally scalable“
- Eric Redmond, Jim R. Wilson, Seven Databases in Seven Weeks, Pragmatic Bookshelf
- Martin Fowler, NoSQL distilled, Addison-Wesley