



**Università
degli Studi
di Palermo**



Introduzione a MongoDB

CORSO DI BIG DATA
a.a. 2022/2023

Prof. Roberto Pirrone

Sommario

- Modello dei dati
- Interazione base con MongoDB
- Caratteristiche generali
 - Indicizzazione
 - Aggregazione e Map-Reduce
- Caratteristiche architetturali
 - Gestione della replicazione e sharding
 - GridFS
- Modellazione del database

Modello dei dati

- Documenti (max 16 MB) in formato BSON (Binary JSON) fatti da coppie chiave-valore
- Ogni documento si trova all'interno di una «collezione»
- Le collezioni hanno in comune le strutture degli indici e costituiscono un database
- I documenti *non hanno* struttura uniforme

MongoDB	RDBMS SQL
Database	Database
Collezione	Tabella
Documento	Record

Modello dei dati

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

Modello dei dati

- Tipi BSON

- Si possono effettuare query usando l'operatore \$type

```
db.test.insertOne( {ts: new Timestamp()})
db.test.find( {"ts": { $type: "timestamp" }})

{ "_id" :
ObjectId( "542c2b97bac0595474108b48" ), "ts"
: Timestamp(1412180887, 1) }
```

Type	Number	Alias
Double	1	"double"
String	2	"string"
Object	3	"object"
Array	4	"array"
Binary data	5	"binData"
Undefined	6	"undefined"
ObjectId	7	"objectId"
Boolean	8	"bool"
Date	9	"date"
Null	10	"null"
Regular Expression	11	"regex"
DBPointer	12	"dbPointer"
JavaScript	13	"javascript"
Symbol	14	"symbol"
JavaScript (with scope)	15	"javascriptWithScope"
32-bit integer	16	"int"
Timestamp	17	"timestamp"
64-bit integer	18	"long"
Decimal128	19	"decimal"
Min key	-1	"minKey"
Max key	127	"maxKey"

Modello dei dati

- Attributo “_id”:
 - Sempre presente in un documento perché è la chiave primaria
 - È immutabile e può essere di qualsiasi tipo non array
 - Il tipo di default è ObjectId
 - Legato al tempo di creazione del documento

Interazione con MongoDB

- Una volta invocato `mongod` o `mongos` da linea di comando si può interagire da shell invocando `mongo`
 - Comandi di shell per gestire i database e avere help
 - Uso di metodi Javascript della API mongoDB:
 - Funzioni di libreria per creare tipi di dati BSON, connettersi a un database remoto, interagire con il SO
 - API vera e propria che usa le classi `db`, `rs` e `sh` per database, replica set e shard
 - Uso di «comandi mongo» in forma di documenti attraverso `db.runCommand()` ovvero `db.adminCommand()`

Interazione con MongoDB

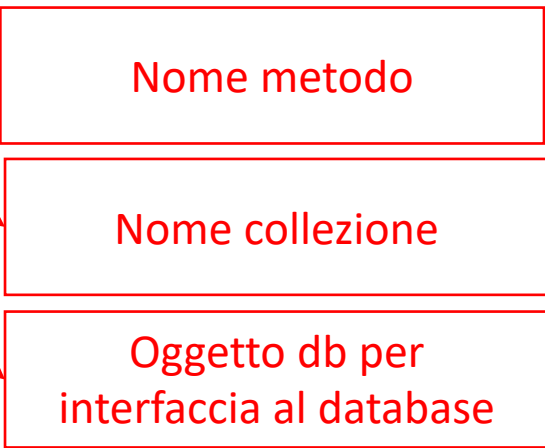
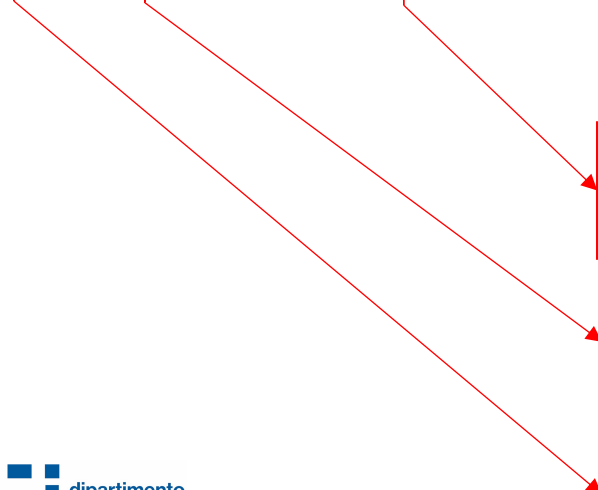
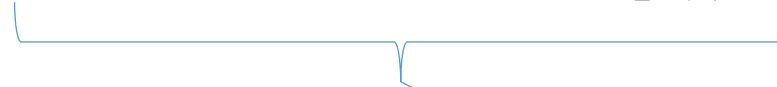
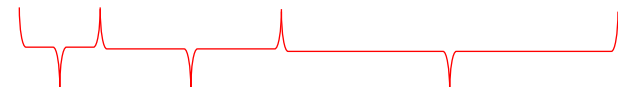
- Comandi di shell principali
 - `show dbs` → elenco dei database
 - `use <nome_database>` → seleziona il database da usare
 - `show collections` → elenco delle collezioni nel database
 - `help` → aiuto generale
 - `db./rs./sh.help()` → aiuto sulle classi db, rs e sh
 - `db.<nome_collezione>.help()` → aiuto sui metodi di collezione

Interazione con MongoDB

- Esempio di chiamata di un metodo Javascript di shell

Metodo Javascript di shell

```
db.test.insertOne( { ts: new Timestamp() } )
```



Documento passato come argomento
(può rappresentare un comando, una
configurazione o veri e propri dati)

Interazione con MongoDB

- Esecuzione di un comando (per es. creazione di un utente)

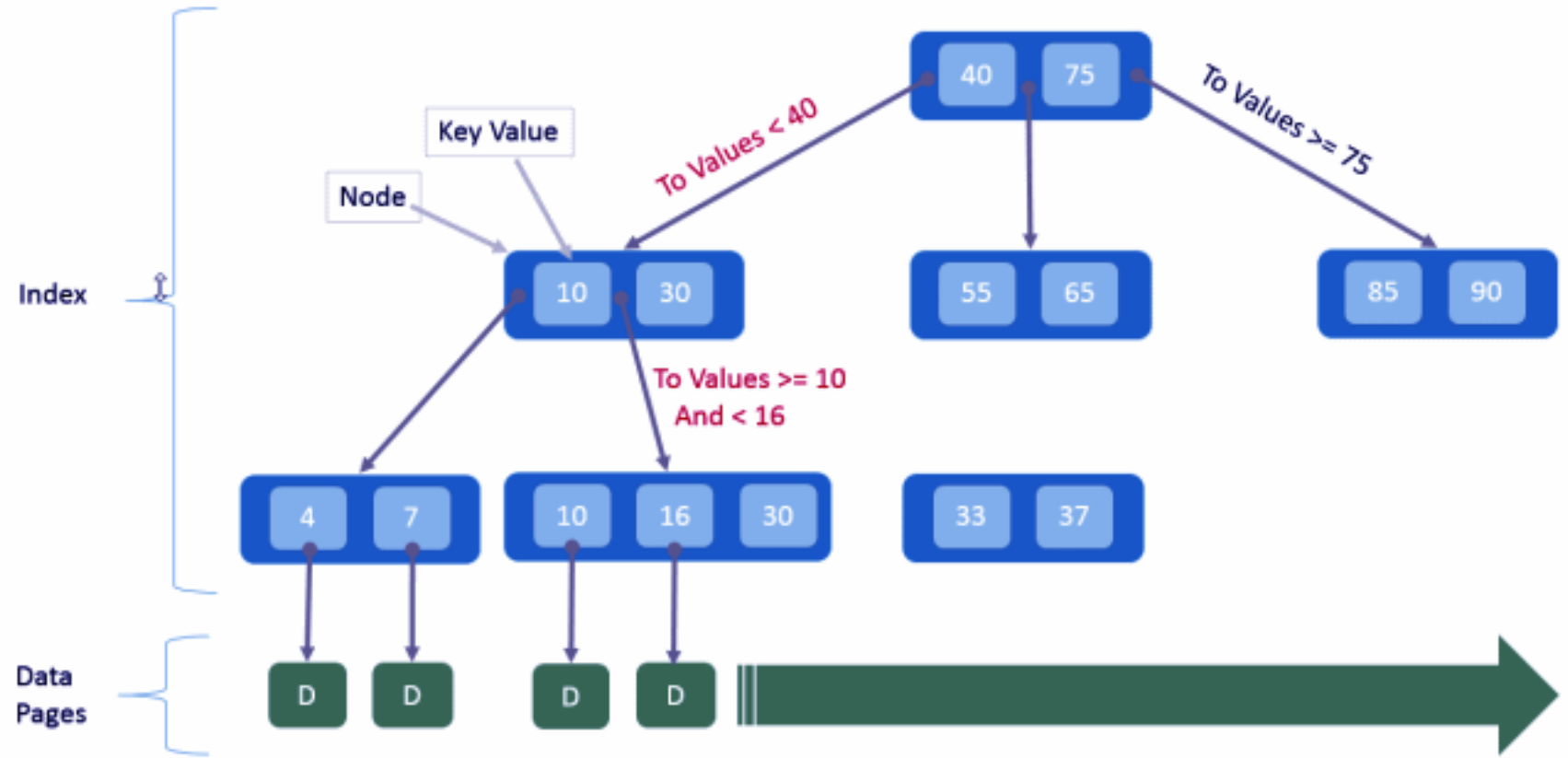
```
db.adminCommand(  
  {  
    createUser: "bruce",  
    pwd: passwordPrompt(), // or <cleartext password>  
    roles: [  
      { role: "dbOwner", db: "admin" }  
    ]  
  }  
)
```

Caratteristiche generali

- Indicizzazione

- Fa uso dei binary tree
- Si possono creare indici singoli e multipli su qualunque campo
- Usati per rendere efficienti le query
- id è l'indice di default

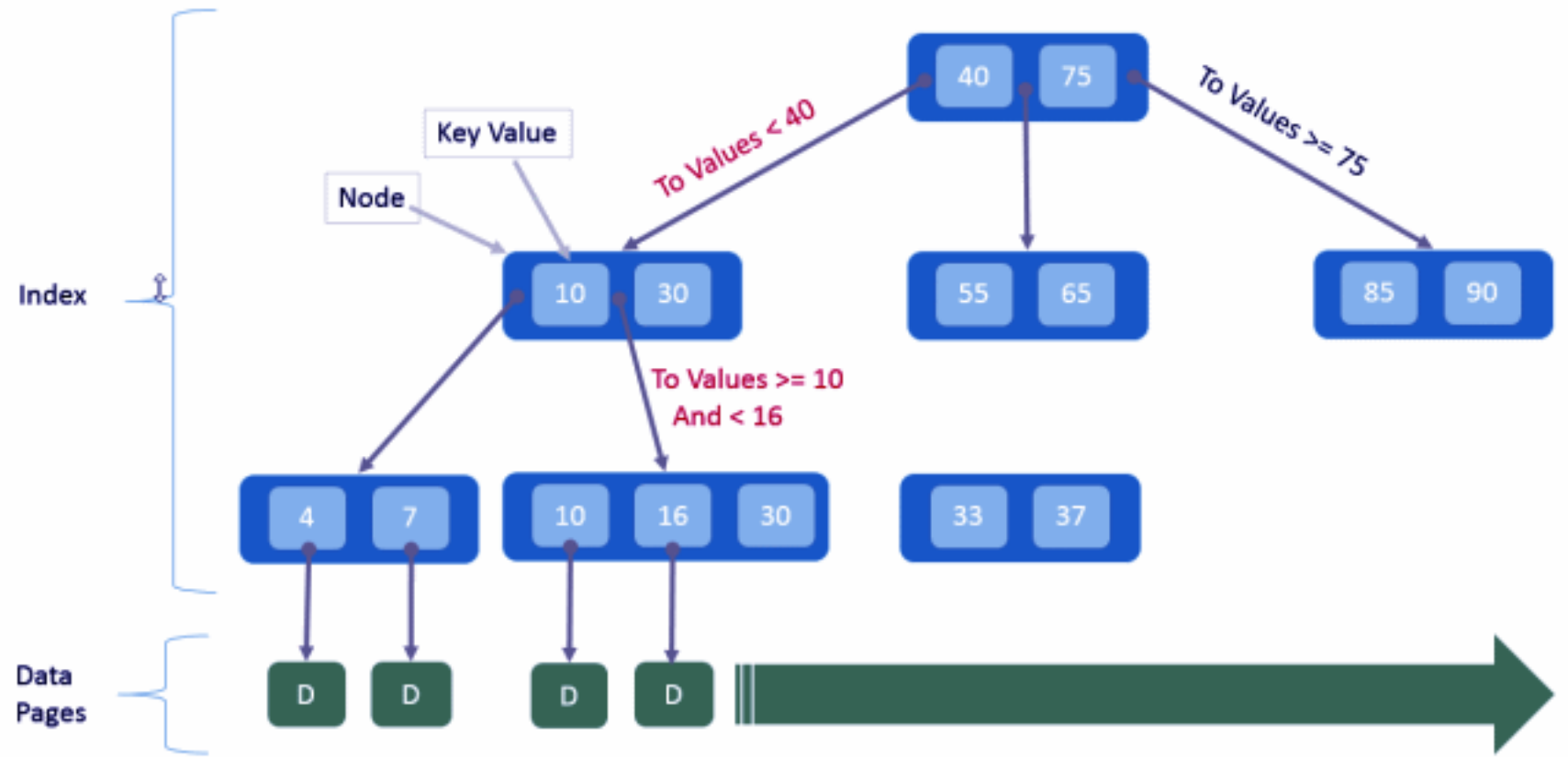
B-Tree Layout



Caratteristiche generali

- Indicizzazione
 - Indici di tipo
 - text per indicizzare testi
 - 2d per dati rappresentati come punti su un piano
 - 2dsphere per dati geospaziali

B-Tree Layout



Caratteristiche generali

- Aggregazione
 - Computazione che agisce su più record e ritorna un solo risultato
 - Pipeline di aggregazione
 - Map-Reduce
 - Aggregazione single purpose

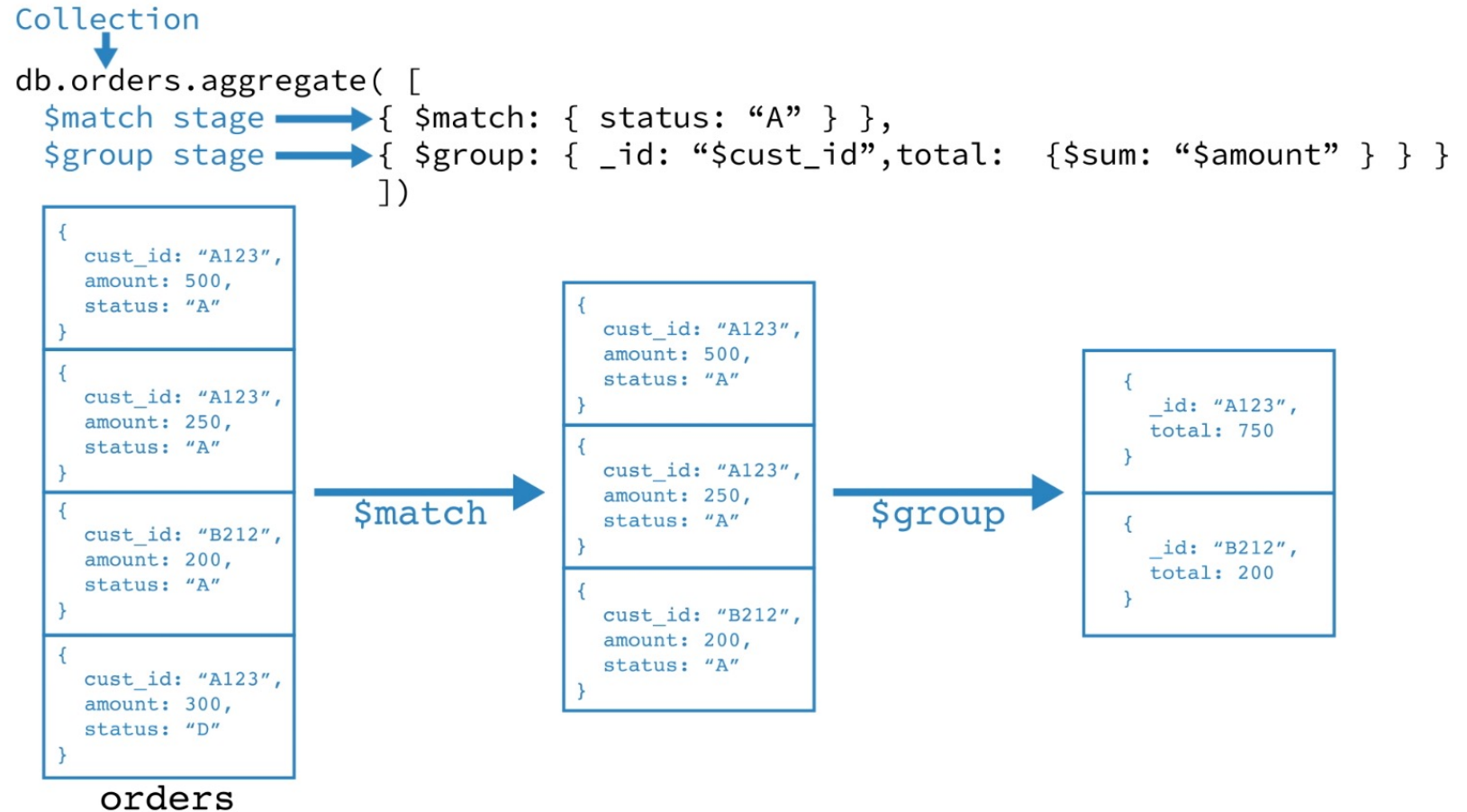


Caratteristiche generali

- Pipeline di aggregazione

- Utilizza una sequenza di operatori mongo (nella forma `$<nome_op>`) per eseguire operazioni di

- Proiezione
- Matching
- Raggruppamento
- Ordinamento
- Selezioni di singoli documenti



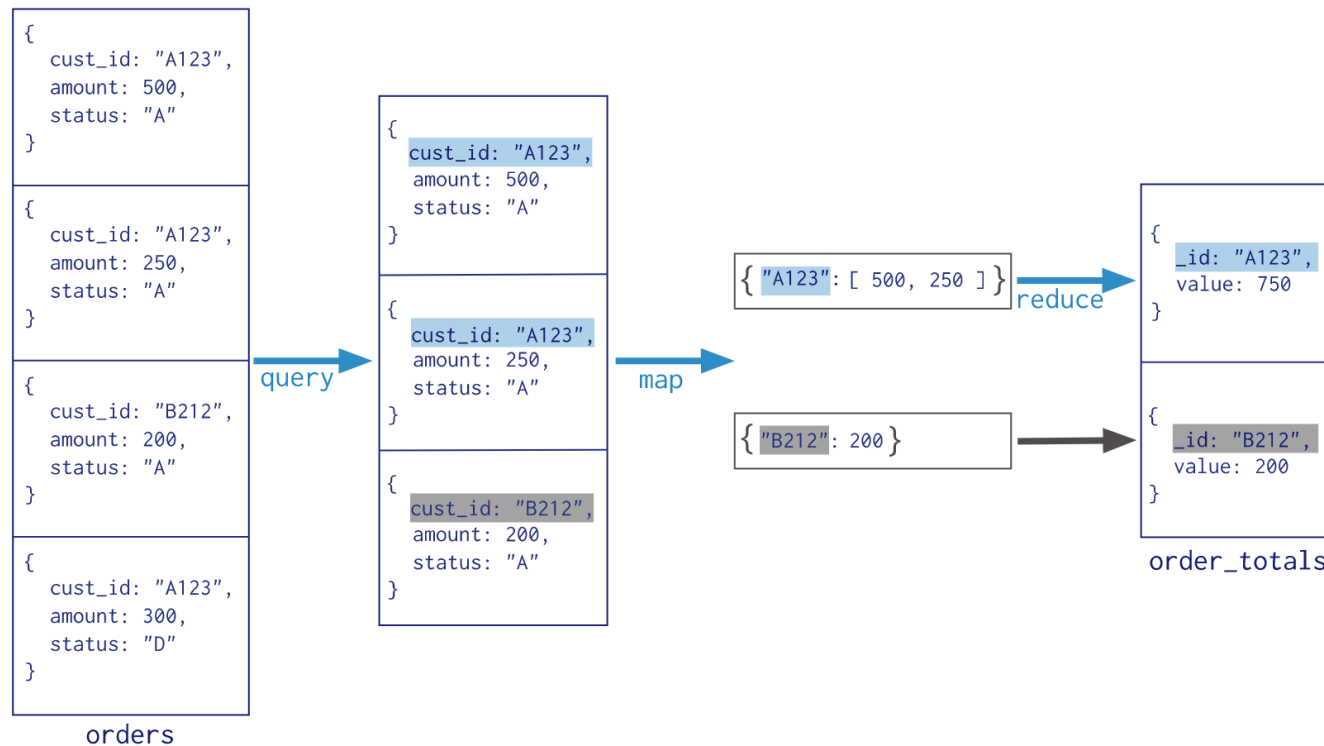
Caratteristiche generali

- Map-Reduce
 - Apposita interfaccia che consente di specificare:
 - La funzione di map
 - La funzione di reduce
 - La query di selezione dei dati dalla collezione di partenza
 - Il formato del output
 - Criteri opzionali di ordinamento
 - Limite al numero dei risultati

Caratteristiche generali

- Map-Reduce

Collection
↓
db.orders.mapReduce(
 map → function() { emit(this.cust_id, this.amount); },
 reduce → function(key, values) { return Array.sum(values) },
 query → { query: { status: "A" },
 output → out: "order_totals"
 }
)



Caratteristiche generali

- Map-Reduce

```
{  
_id: ObjectId("50a8240b927d5d8b5891743c"),  
cust_id: "a123",  
ord_date: new Date("Jan 04, 2019"),  
status: 'A',  
price: 25,  
items: [ { sku: "m", qty: 5, price: 2.5 },  
{ sku: "n", qty: 5, price: 2.5 } ]  
}
```

Caratteristiche generali

- Map-Reduce

```
var mapFunction = function() {  
    for (var idx = 0; idx < this.items.length; idx++) {  
        var key = this.items[idx].sku;  
        var value = {  
            count: 1,  
            qty: this.items[idx].qty  
        };  
        emit(key, value);  
    }  
};
```

Caratteristiche generali

- Map-Reduce

```
var reduceFunction = function(key, ObjVals) {  
    reducedVal = { count: 0, qty: 0 };  
  
    for (var idx = 0; idx < ObjVals.length; idx++) {  
        reducedVal.count += ObjVals[idx].count;  
        reducedVal.qty += ObjVals[idx].qty;  
    }  
  
    return reducedVal;  
};
```

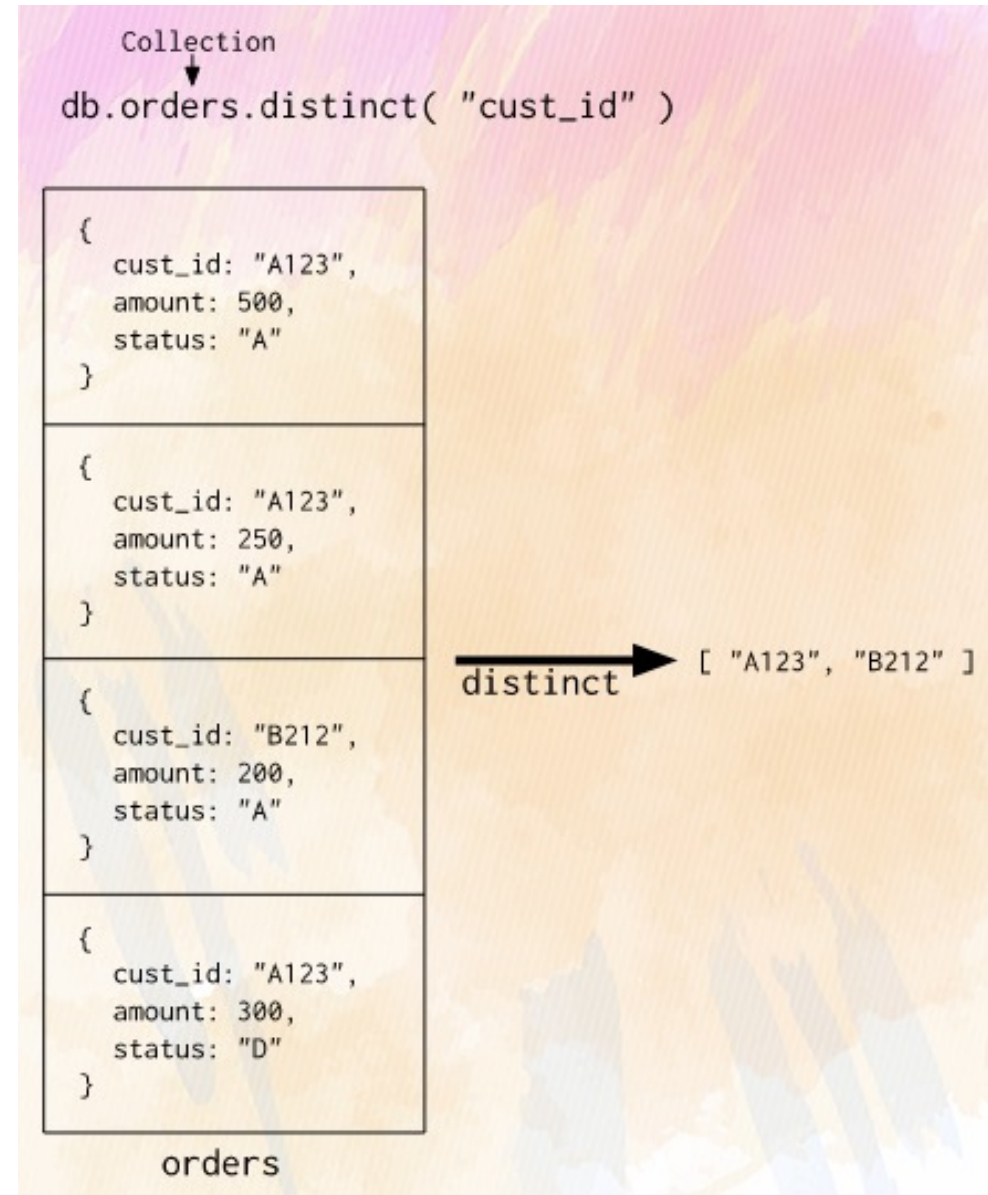
Caratteristiche generali

- Map-Reduce

```
var finalizeFunction = function (keys, reducedVal) {  
    reducedVal.avg = reducedVal.qty/reducedVal.count;  
    return reducedVal;  
};  
db.examples.mapReduce( mapFunction, reduceFunction,  
    {  
    query: { ord_date: { $gt: new Date('26/01/2019') } },  
    out: { merge: "map_reduce_example" },  
    finalize: finalizeFunction  
    }  
)
```

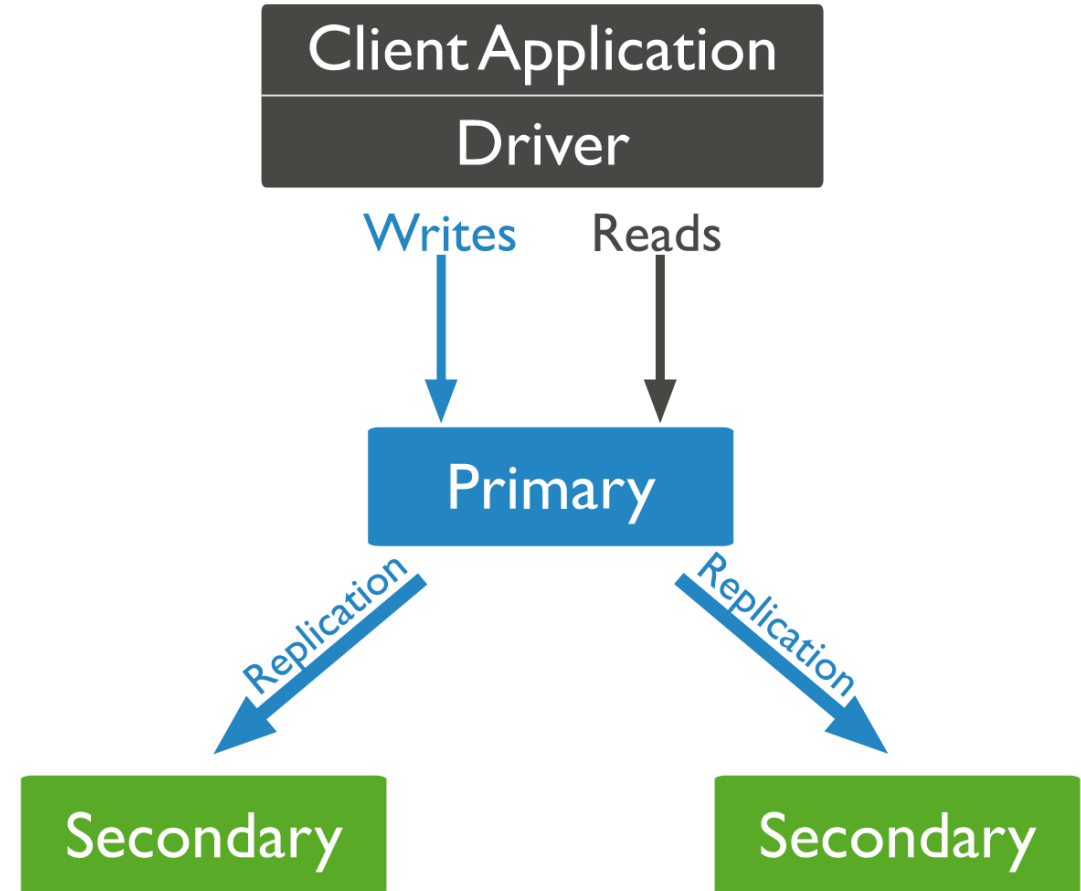
Caratteristiche generali

- Aggregazione single purpose
 - Singoli metodi o comandi di database che eseguono operazioni su collezioni
 - distinct
 - count



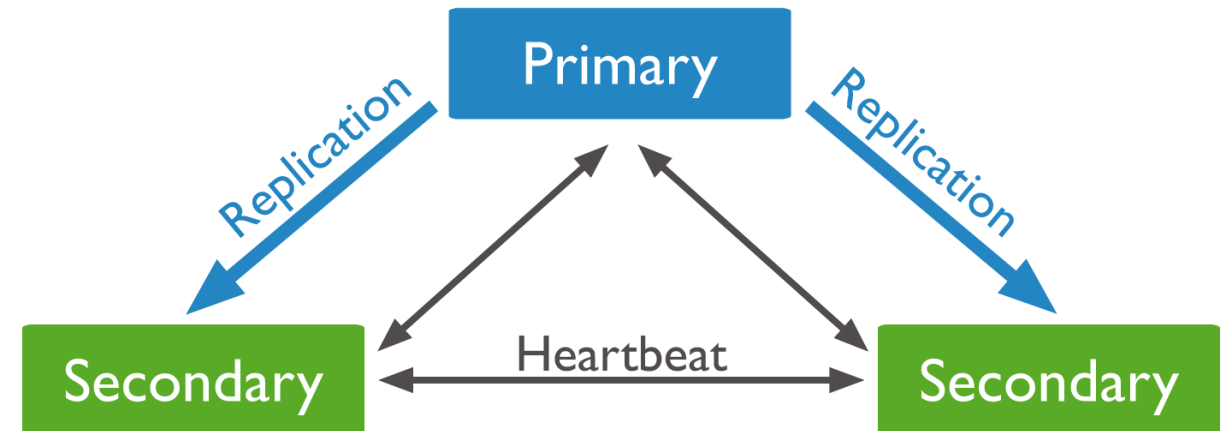
Caratteristiche architetture

- Replicazione
 - Differenti istanze mongod su differenti macchine vengono configurate come nodi primari o secondari
 - Si parla globalmente di «replica set»
 - Si possono configurare dalla shell di mongo ovvero come comandi del database



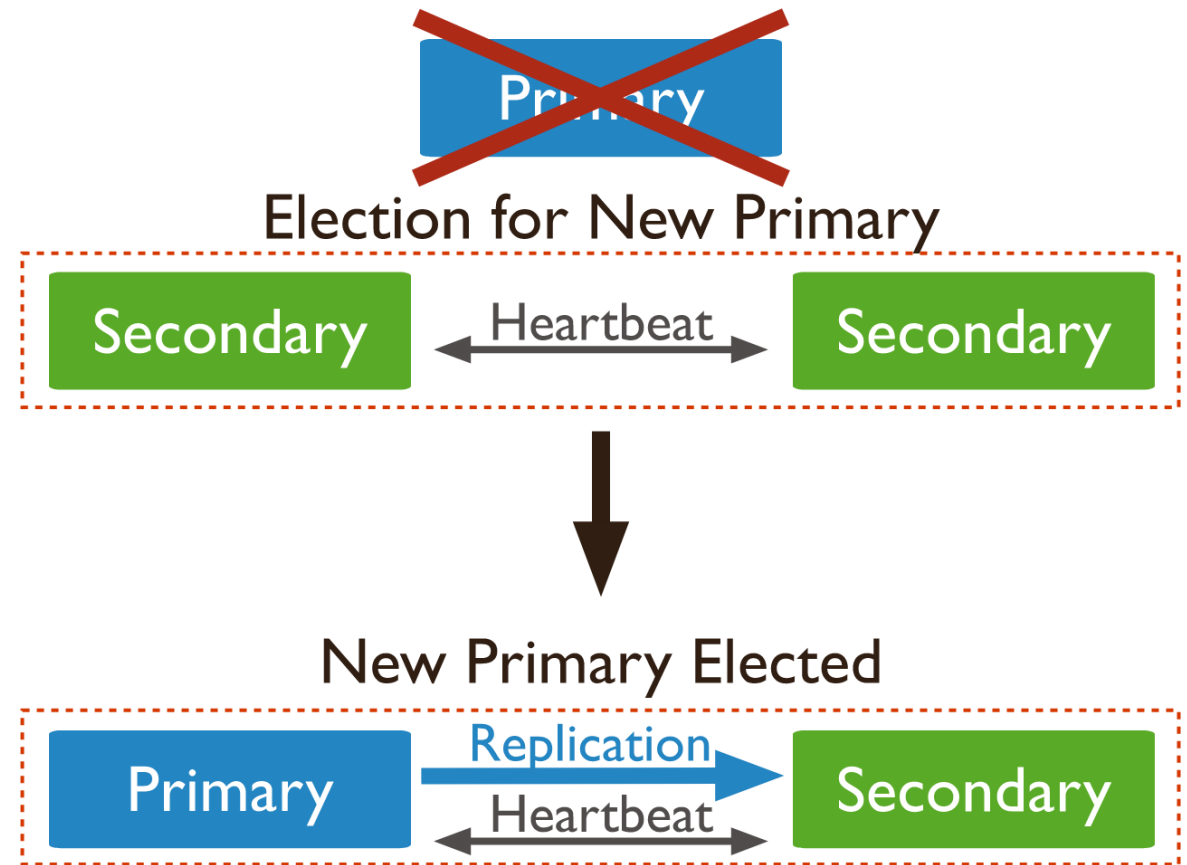
Caratteristiche architetture

- Replicazione
 - I secondari replicano il log delle operazioni del primario e le applicano ai propri dati
 - Sincronizzazione continua tramite heartbeat



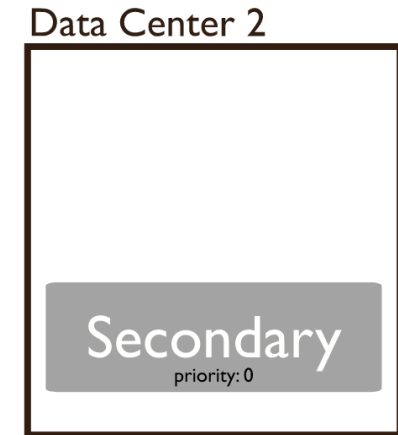
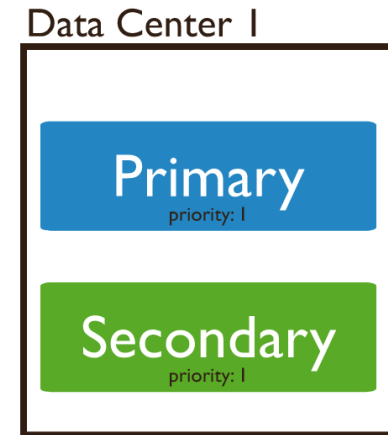
Caratteristiche architeturali

- Replicazione
 - In caso di fault del primario un secondario può transitoriamente «essere eletto» primario
 - Gli altri secondari replicano il suo comportamento



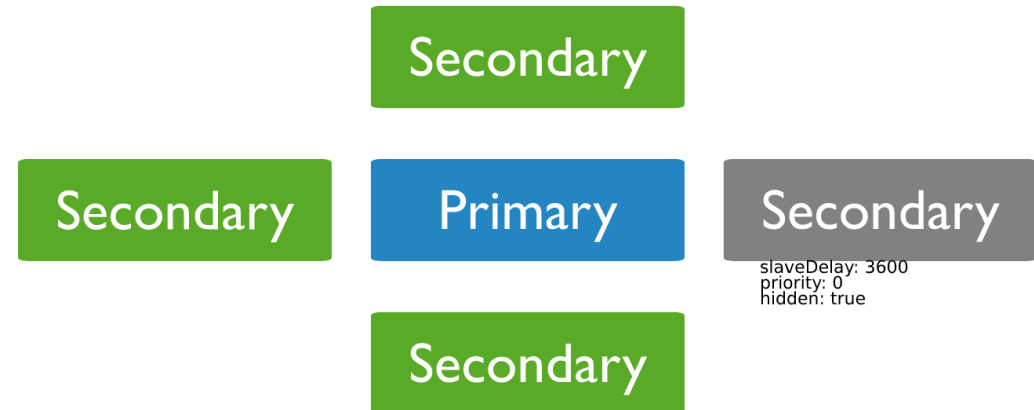
Caratteristiche architetture

- Replicazione
 - Un secondario può avere «priorità 0»
 - In questo caso non può essere eletto primario



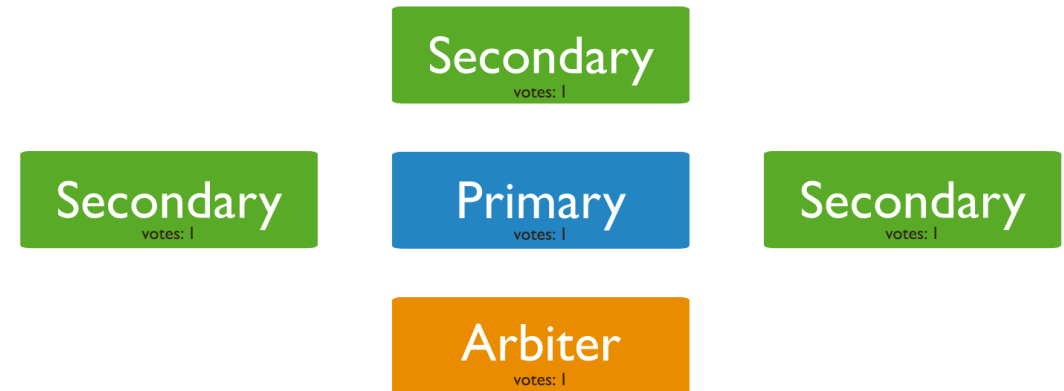
Caratteristiche architeturali

- Replicazione
 - Secondari nascosti: copie del primario a priorità 0, invisibili ai client che «possono» votare per il primario
 - Secondari delayed: sono nascosti e riportano una replica di un set di nodi aggiornata con un dato ritardo di tempo
 - Ha funzioni di backup



Caratteristiche architetture

- Replicazione
 - Arbitri: non mantengono copie dei dati e non possono divenire primari
 - Hanno un voto e servono a mantenere dispari la maggioranza dei voti per garantire l'elezione del primario



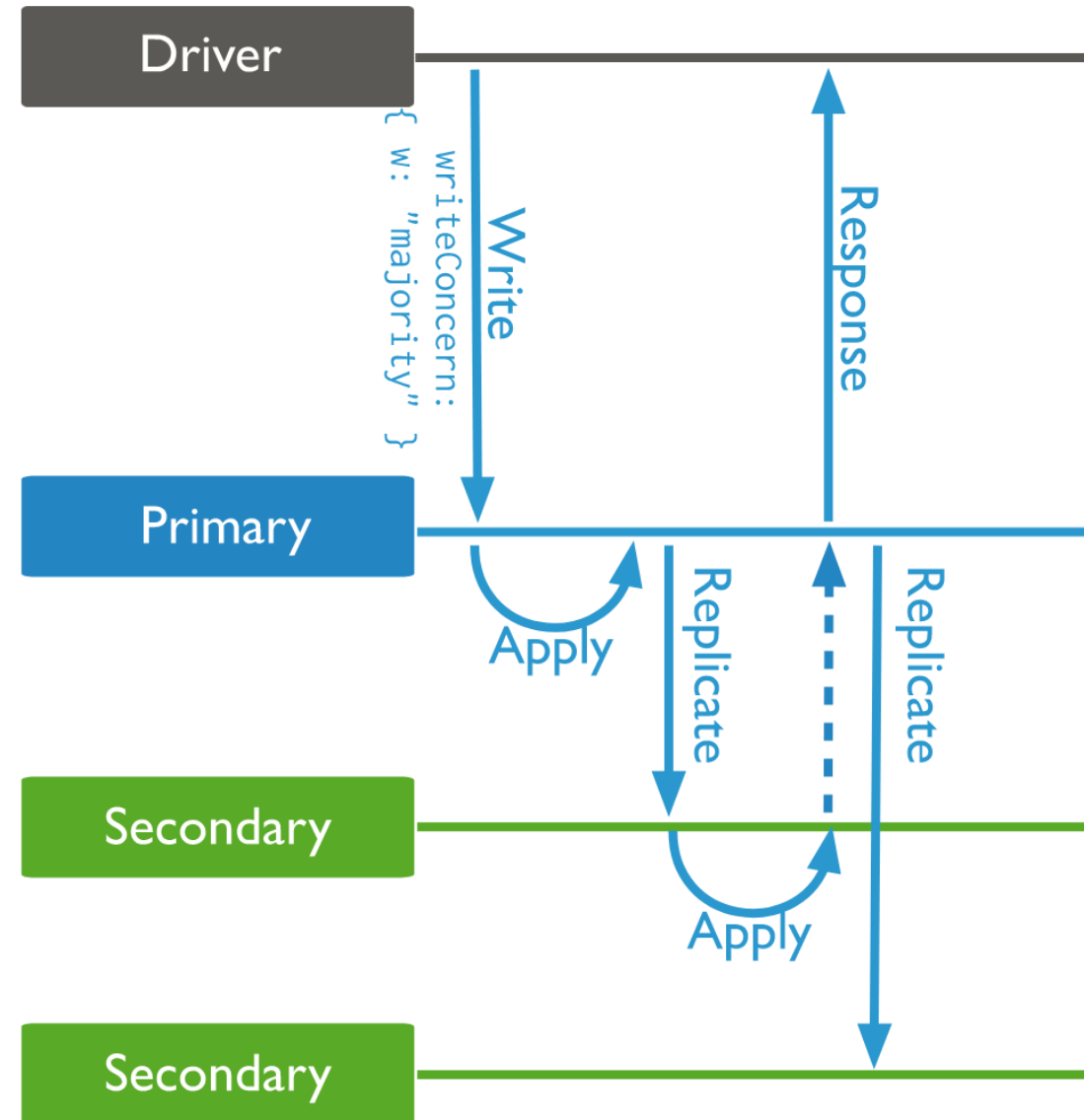
Caratteristiche generali

- Replicazione

- Il «write concern» stabilisce il criterio con cui si ottiene il feedback sulla scrittura nel replica set

```
db.products.insert(  
  { item: "envelopes", qty : 100,  
    type: "Clasp" },  
  { writeConcern: { w: "majority",  
                    j: true, wtimeout: 5000 } } )
```

journaling



Caratteristiche architetture

- Replicazione

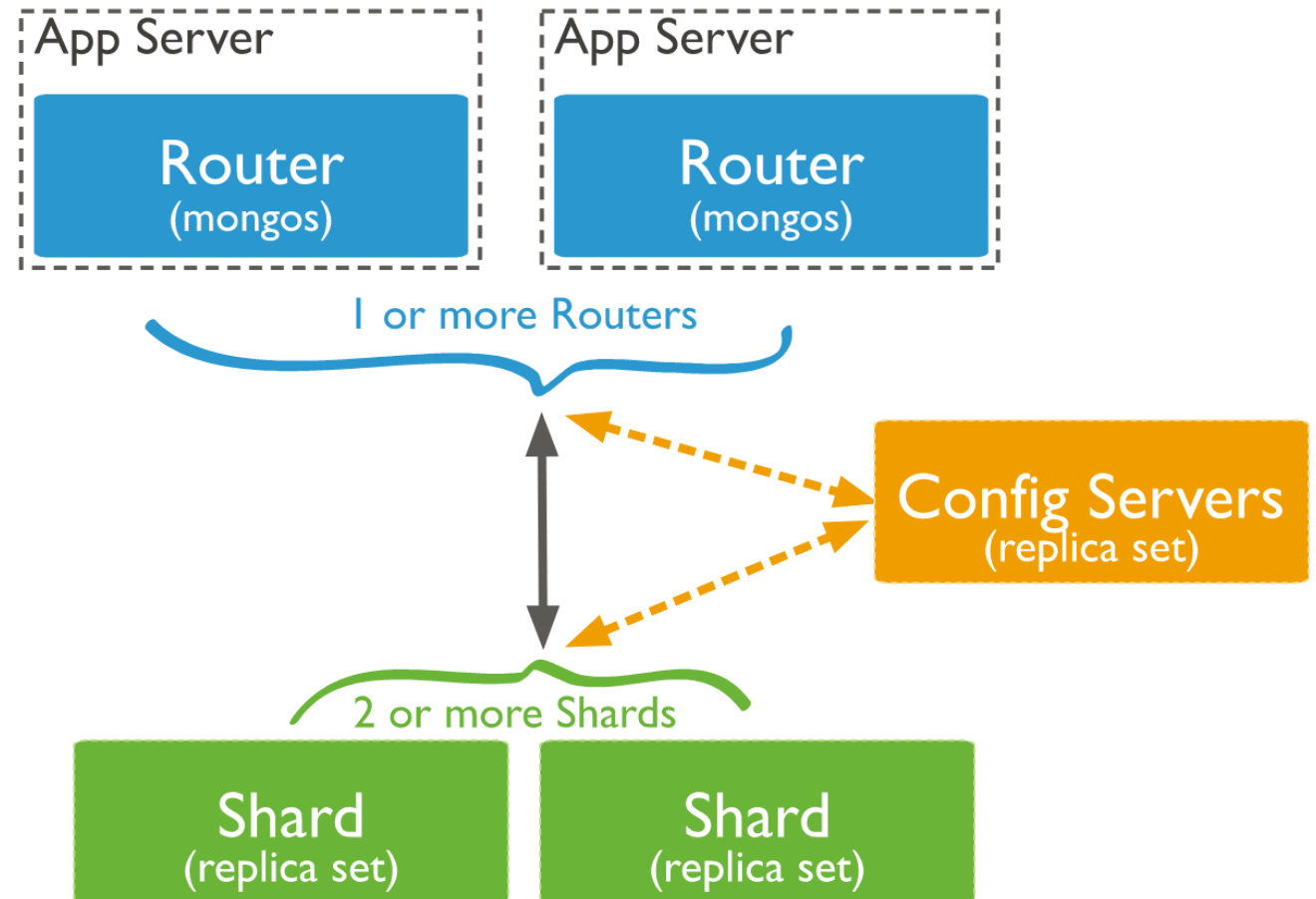
- la «read preference» stabilisce il criterio con cui si legge da un nodo:
 - `primary`, `primaryPreferred`, `secondary`, `secondaryPreferred`, `nearest`
- Il «read concern» stabilisce il criterio di consistenza

```
db.collection.find().readConcern("majority")
```

se anche il write concern è impostato a `majority`, implica una consistenza del tipo Read Your Own Writes

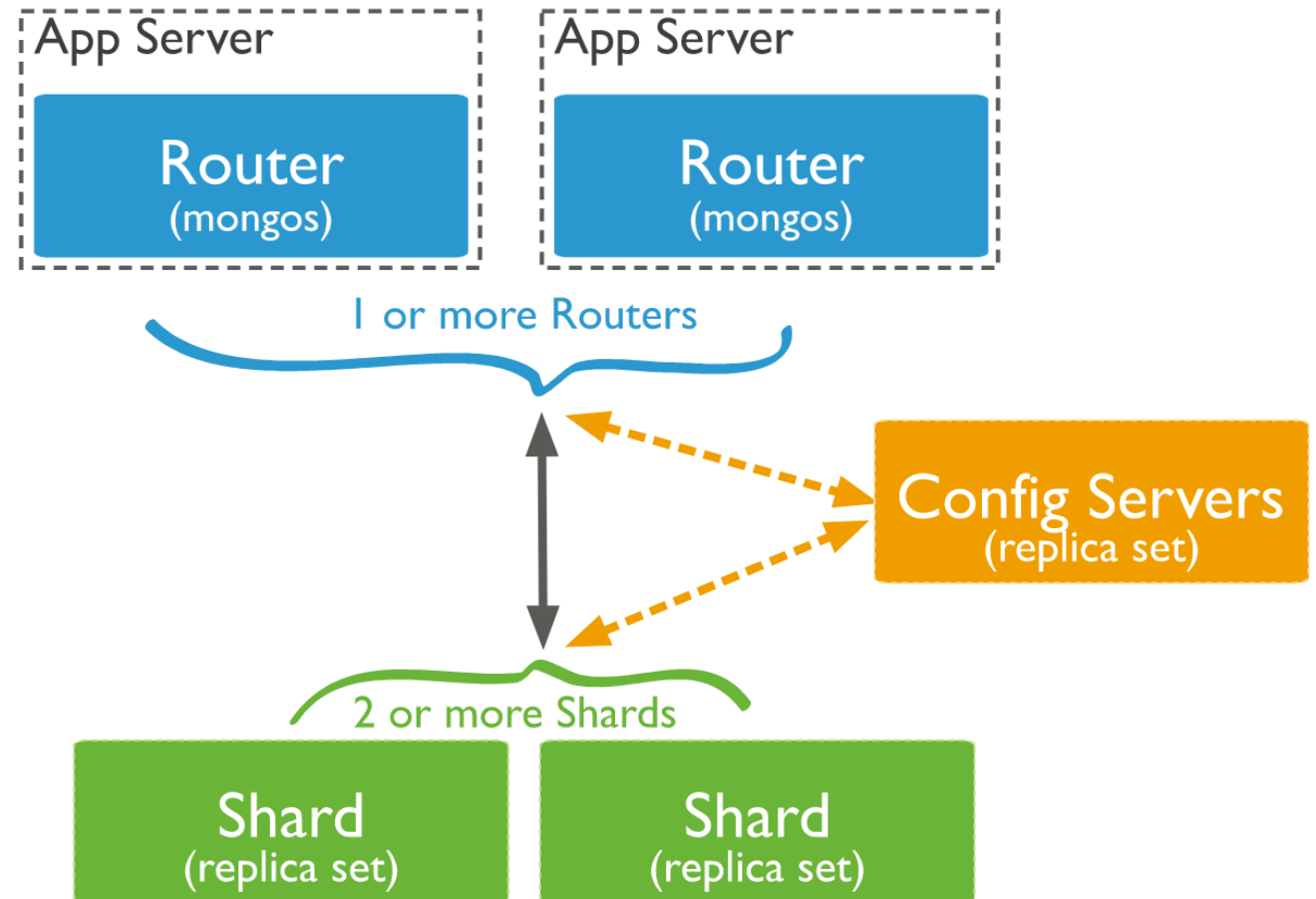
Caratteristiche architetture

- Sharding
 - Suddivisione dei dati in «frammenti» che vengono distribuiti in un cluster
 - Possono essere configurati anche come un «replica set» usando la replicazione
 - Il router è mongos e non mongod



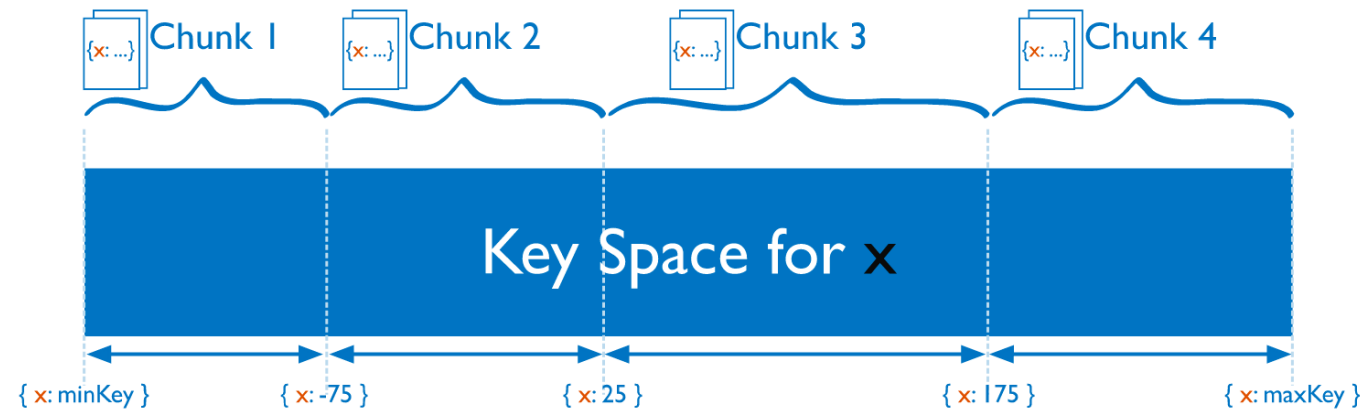
Caratteristiche architetture

- Sharding
 - Più router e più shard
 - I server di configurazione contengono i metadati e le impostazioni di configurazione per il cluster



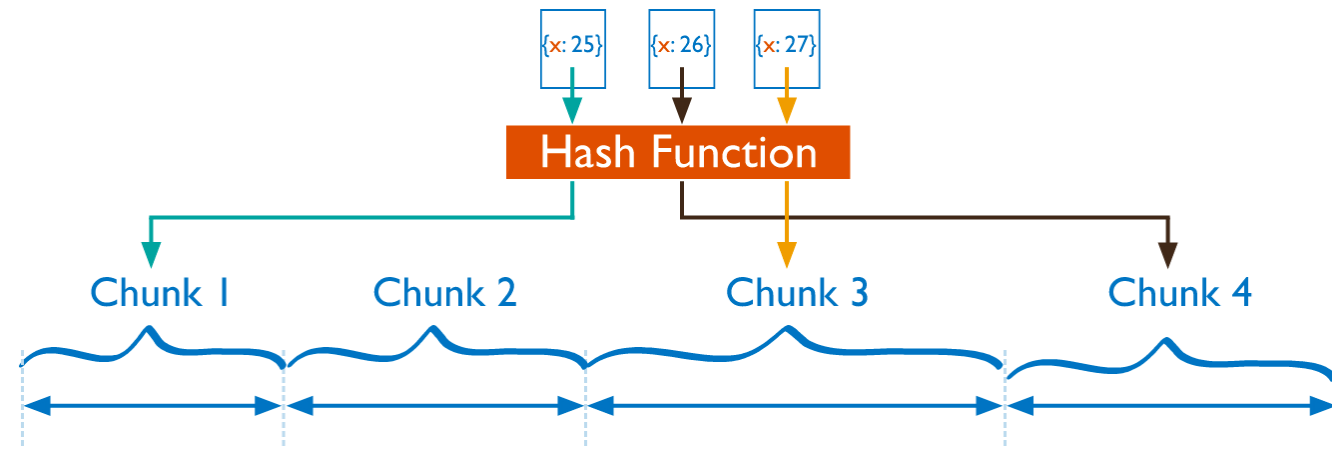
Caratteristiche architetture

- Sharding
 - La «shard key» è la chiave che gestisce lo sharding in una collezione
 - mongoDB cerca di distribuire i valori della chiave uniformemente nei vari shard
 - *Ranged sharding*
 - La collezione *deve avere* un indice che contenga la shard key



Caratteristiche architeturali

- Sharding
 - Una buona shard key deve
 - Cambiare poco
 - Avere un range di valori che consenta una ampia distribuzione dei documenti negli shard
 - Non avere particolari valori che occorrono troppo frequentemente → lo shard si carica di troppi documenti



Hashed sharding per gestire la distribuzione uniforme
Dei valori della chiave

Caratteristiche architetture

- Sharding

```
sh.addShard("Server1:27017")
```

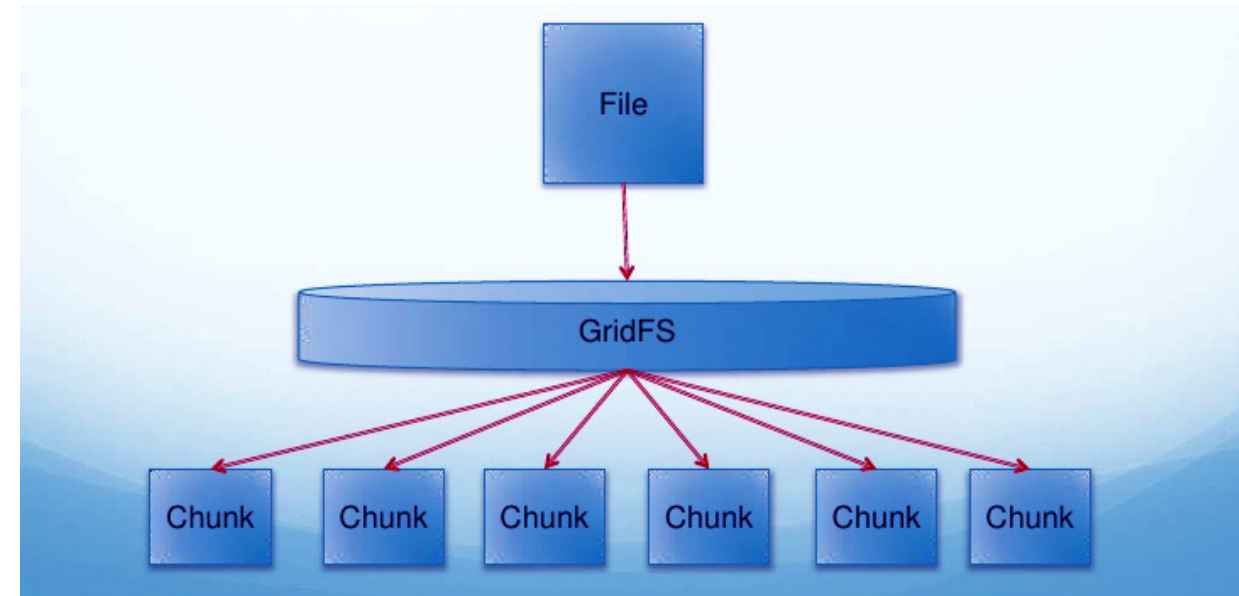
```
sh.addShard("Server2:27017")
```

```
sh.enableSharding(Studentdb)
```

```
sh.shardCollection("db.Student", {"Studentid": 1, "StudentName": 1})
```

Caratteristiche architetture

- GridFS
 - gestione di grandi file attraverso «chunk» di 256KB
 - L'interfaccia è data da una coppia apposita di collezioni che possono essere accedute da ogni database per creare i chunk e interrogare i dati così organizzati
 - `fs.files`
 - `fs.chunks`



Modellazione del database

- E' possibile modellare uno schema relazionale all'interno di una collezione
- Si adottano due schemi
 - Embedding
 - Reference o linking

Modellazione del database

- Embedding
 - Relazione 1-1
 - Relazione 1-N se si tratta di un «array» di documenti
 - Denormalizzata



Modellazione del database

- Reference

- Relazioni 1-1/1-N

- Normalizzata:
implica
automaticamente
1-N se lo stesso
_id lato 1 si
ritrova dal lato N

user document

```
{  
  _id: <ObjectId1>,  
  username: "123xyz"  
}
```

contact document

```
{  
  _id: <ObjectId2>,  
  user_id: <ObjectId1>,  
  phone: "123-456-7890",  
  email: "xyz@example.com"  
}
```

access document

```
{  
  _id: <ObjectId3>,  
  user_id: <ObjectId1>,  
  level: 5,  
  group: "dev"  
}
```

Modellazione del database

- La relazione N-N si può facilmente ottenere replicando gli «_id» da entrambe le parti
- Non c'è un supporto per «attributi di relazione»
 - Contro la filosofia Schema-On-Write
 - L'applicazione recupera e struttura i dati per suo conto