

## **Stima dei costi database non relazionali**

Calcolo e stima dei costi di elaborazione di diverse query su attributi indicizzati  
Database non relazionali in questione MongoDB e CouchDB

Presentata da:  
**Matteo Santoro**  
Matricola 0000881608

Relatore:  
**Prof. Alessandra Lumini**  
Correlatore:  
**Prof. Alessandra Lumini**

# Parole chiave

Parola chiave 1

Parola chiave 2

*Dedica*



## **Sommario**

I database sono diventati una parte importante del quotidiano, ogni tipo di applicazione o software che siamo abituati ad utilizzare ha l'esigenza di prendere, processare e conservare diversi tipi di dati, e' quindi fondamentale poter ottimizzare al meglio queste operazioni, all'interno della tesi che presentero' andro' ad analizzare e stimare alcuni possibili metodi e condurre un'analisi comparativa anche attraverso diversi sistemi di database. Io e la Prof.ssa Lumini abbiamo deciso di utilizzare due database Document-Based non relazionali, MongoDB e CouchDB ed un database di tipo relazionale, Oracle. La finalita' di questa tesi e' arrivare a valutare il miglior piano d'accesso possibile e la modellazione dei dati piu' efficiente.



# Indice

<b>Introduzione</b>	<b>IX</b>
1 Differenze tra relational/non-relational DB . . . . .	IX
2 Il nostro caso di studio . . . . .	IX
3 Modellazione dei dati . . . . .	XII
<b>Costi</b>	<b>XV</b>
1 Calcolo dei costi . . . . .	XV
1.1 NL . . . . .	XV
1.2 g . . . . .	XV
1.3 h . . . . .	XV
1.4 NP . . . . .	XV
1.5 Costo del nested loop . . . . .	XV
1.6 Indice clustered . . . . .	XVI
1.7 Indice unclustered . . . . .	XVI
<b>MongoDb</b>	<b>XIX</b>
1 Caratteristiche . . . . .	XIX
2 Organizzazione dei dati . . . . .	XIX
2.1 Creazione collezioni . . . . .	XIX
2.2 Indici . . . . .	XXVII
<b>Oracle</b>	<b>XXXI</b>
1 Raccolta dati . . . . .	XXXI
<b>CouchDb</b>	<b>XXXIII</b>
1 Caratteristiche . . . . .	XXXIII
2 Popolamento . . . . .	XXXIII
3 Indici . . . . .	XXXIV





# Introduzione

## 1 Differenze tra relational/non-relational DB

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Suspendisse et ex vehicula, interdum mi eu, auctor augue. Suspendisse vel sagittis urna. In vitae ligula ipsum. Vivamus mattis neque efficitur, gravida purus facilisis, rhoncus felis. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Nunc bibendum urna porta quam congue ornare. Fusce eget consequat libero. Donec varius justo vel libero malesuada, sit amet dignissim ex tincidunt. In pharetra vestibulum lacus quis laoreet. Donec vel laoreet ex, sed facilisis massa. Mauris commodo velit est. Maecenas non sem elementum, faucibus velit et, bibendum mi. Integer laoreet ex et eros accumsan, nec dapibus nisi vestibulum. Donec sed nunc quis mi gravida congue non vel tortor. Interdum et malesuada fames ac ante ipsum primis in faucibus.

Nullam non pharetra arcu. Donec eget velit elit. Pellentesque sed tortor sodales neque tristique rutrum. Vivamus et ex dolor. Vestibulum lacinia augue sit amet libero mattis pellentesque. In sed congue ante, quis tempus neque. Vestibulum semper eu sapien et vestibulum.

In ac erat ullamcorper, ultricies dolor sit amet, tempus neque. Pellentesque quam erat, ornare ac justo at, cursus luctus ex. Pellentesque at turpis blandit, elementum ex ac, fringilla nibh. Etiam et tincidunt lacus. Integer mattis mi sit amet faucibus rutrum. Sed at nisi commodo, ultricies purus a, tempus lacus. Mauris accumsan enim nisi, in tempor velit pharetra eu. Sed eu turpis et ante sagittis sodales. Duis a tellus id risus ultricies accumsan. Vestibulum bibendum in sapien sit amet rhoncus. Fusce aliquam, metus vel efficitur pulvinar, nibh lacus ultricies nisl, nec rhoncus elit ligula vitae risus. Mauris bibendum eget erat non rutrum. Curabitur in ligula eget lectus facilisis molestie sed sed neque. Vestibulum eu faucibus augue, a luctus augue. Nam lobortis massa non lorem condimentum vehicula. Aliquam efficitur cursus neque, efficitur placerat libero tincidunt non. Nullam.

## 2 Il nostro caso di studio

Durante la progettazione si e' deciso di utilizzare una relazione per poter testare i diversi metodi di collezione dei dati. La relazione e' formata da 2 entita' A e B, in un esempio di

un social network l'entita' A sono i Post e l'entita' B i Commenti sotto ogni post, collegate da una relazione 1..N, per ogni A esistono 10 B relativi, ogni Post contiene 10 Commenti.

La modellazione delle entita' e' stata fatta creando 2 triplette di attributi uguali, sulla seconda tripletta al contrario della prima sono stati costruiti degli indici su ogni attributo, questi attributi generici hanno una propria selettivita':

$$\left. \begin{aligned} sel(Att1, Att4) &= \frac{1}{10} \\ sel(Att2, Att5) &= \frac{1}{10.000} \\ sel(Att2, Att6) &= \frac{1}{100} \end{aligned} \right\}$$

Poi un attributo testuale di 100 bytes per poter inserire descrizione.

Il numero di Entita' A e' di  $10^5$  e di conseguenza ci sono  $10^6$  entita' B.

— A (POST) —

```
"_id": 6808,
"AK": 6808,
"A1": 3042,
"A2": 300,
"A3": 8,
"A4": 3042,
"A5": 300,
"A6": 8,
"A7": "Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Fusce lacinia eget arcu et maximus. Ut tempus est sit amet
tortor commodo, sit amet facilisis mi rhoncus. Donec et elit
venenatis, consequat tellus eu, tristique orci. Duis
tristique sem ut nulla ullamcorper, a porta risus efficitur.
Cras sed neque et nisl tincidunt vestibulum. Phasellus
tristique tempor facilisis. Sed facilisis lectus eros, sed
aliquet lacus elementum sed. Integer vel dictum mi.
Maecenas pharetra tempus eros, efficitur mattis erat cursus
in. Nulla sit amet quam velit. Nullam tempus dictum lacus
id porttitor. Vestibulum facilisis pulvinar fermentum.
Ut elementum maximus feugiat. In at mollis leo, eu
facilisis magna. Vestibulum sed nisi ultricies, tincidunt
enim ac, fringilla ex. Phasellus pharetra mollis nisi
a fermentum. In nec faucibus nulla, eget molestie magna.
Vivamus in gravida ex. Aenean scelerisque gravida
ipsum, nec congue enim posuere sit amet. Donec vitae felis
```

id sem congue blandit eget non justo quis."

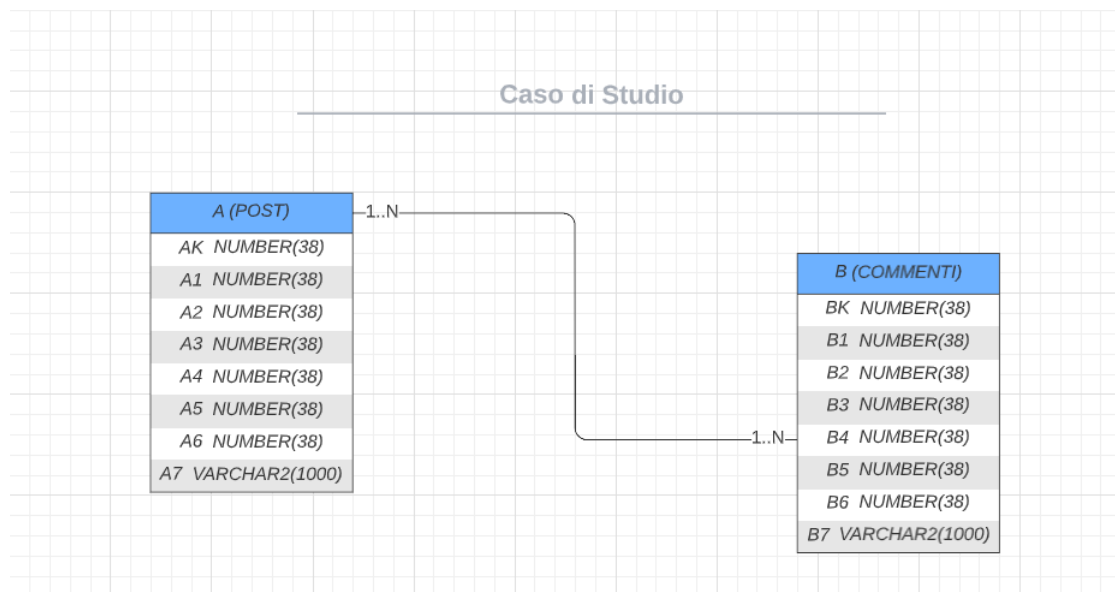
A (POST)

B (COMMENTI)

```
"_id": 73888,  
"BK": 738878,  
"F_AK": 70394,  
"B1": 94207,  
"B2": 176,  
"B3": 3,  
"B4": 94207,  
"B5": 176,  
"B6": 3,  
"B7": "Lorem ipsum dolor sit amet, consectetur adipiscing elit.  
Fusce lacinia eget arcu et maximus. Ut tempus est sit amet  
tortor commodo, sit amet facilisis mi rhoncus. Donec et elit  
venenatis, consequat tellus eu, tristique orci. Duis  
tristique sem ut nulla ullamcorper, a porta risus efficitur.  
Cras sed neque et nisl tincidunt vestibulum. Phasellus  
tristique tempor facilisis. Sed facilisis lectus eros, sed  
aliquet lacus elementum sed. Integer vel dictum mi.  
Maecenas pharetra tempus eros, efficitur mattis erat cursus  
in. Nulla sit amet quam velit. Nullam tempus dictum lacus  
id porttitor. Vestibulum facilisis pulvinar fermentum.  
Ut elementum maximus feugiat. In at mollis leo, eu  
facilisis magna. Vestibulum sed nisi ultricies, tincidunt  
enim ac, fringilla ex. Phasellus pharetra mollis nisi  
a fermentum. In nec faucibus nulla, eget molestie magna.  
Vivamus in gravida ex. Aenean scelerisque gravida  
ipsum, nec congue enim posuere sit amet. Donec vitae felis  
id sem congue blandit eget non justo quis."
```

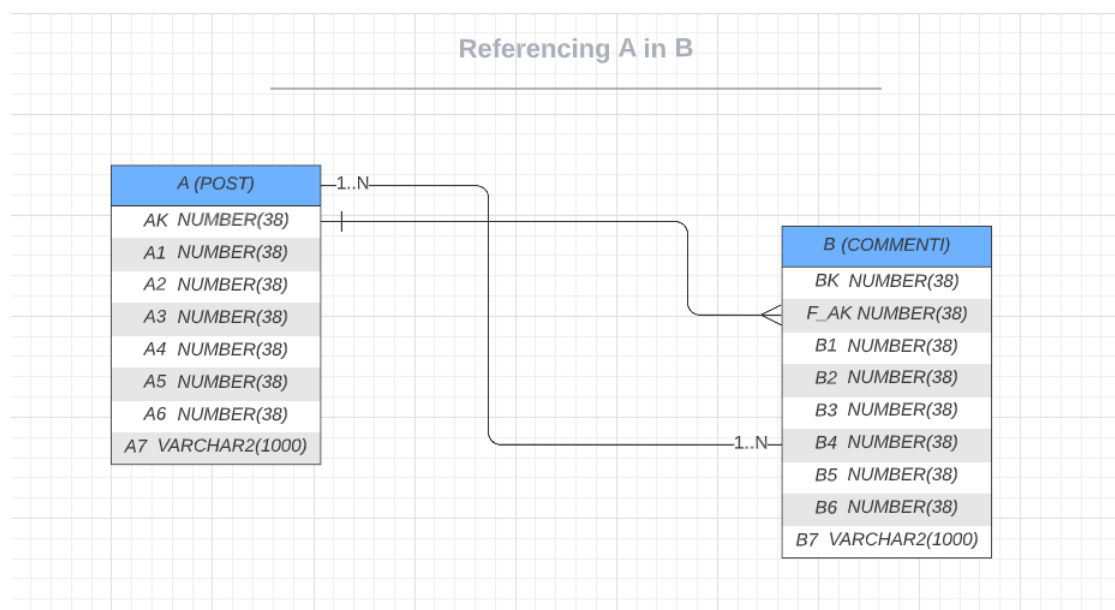
B (COMMENTI)

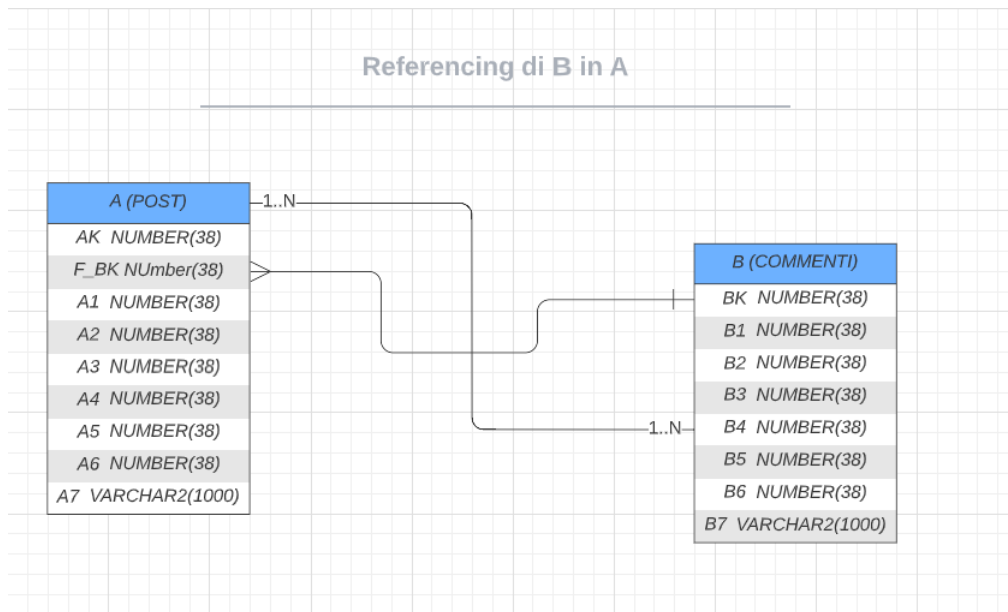
!!!!!!!!! LE FOTO NON VANNO BENE, LE CHIAVI DEVONO ESSERE SEGNALATE  
!!!!!!!!!!!!!!!!!!!!!!!!!!!!



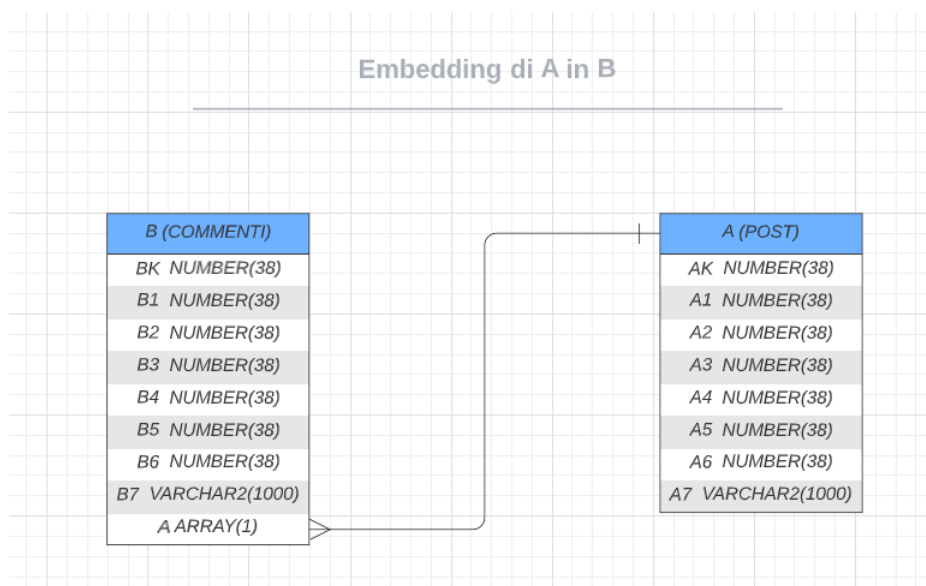
### 3 Modellazione dei dati

La modellazione dei dati ha un ruolo fondamentale nel calcolo dei costi di accesso e quindi costi di esecuzione delle query, le soluzioni dalle quali si e' deciso di di partire sono quelle dell'Embedding e del Referencing. Utilizzando un sistema non relazionale abbiamo necessita' di esprimere la relazione tra A e B in un modo differente, attraverso il referencing colleghiamo le un entita' inserendo un riferimento della seconda e viceversa, per poter poi eseguire ad esempio query di join o voler recuperare tutti i dati dell'entita' referenziata c'e' il bisogno di mantenere all'interno del database la collezione contenente i rimanenti attributi. Es. nell'eventuale soluzione del referencing di B all'interno di A, ci sono due collezioni ossia A con la foreign key relativa (BK) relativa a B e l'intera collezione B.

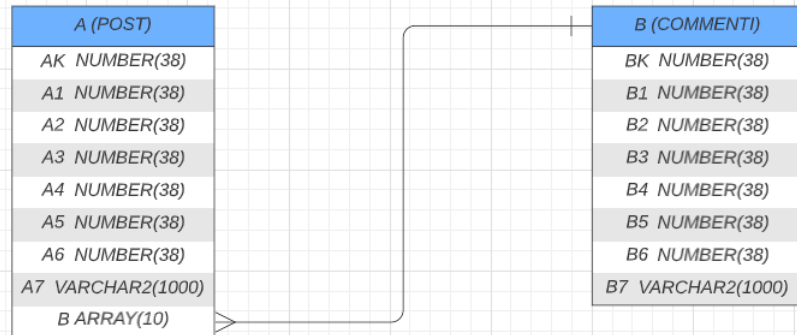




La soluzione embedding consiste nel creare un'unica collezione di documenti che racchiudono al loro interno sia documenti di A che documenti di B.



## Embedding di B in A



# Costi

## 1 Calcolo dei costi

La parte teorica di questo progetto si basa sulla stima di costi di esecuzione delle operazioni sulle differenti opzioni di organizzazione dei documenti, per calcolare cio' abbiamo bisogno di alcune formule che andro qui ad elencare:

### 1.1 NL

$$NL = \left\lfloor \frac{NK * len(k) + NR * len(p)}{D * u} \right\rfloor$$

### 1.2 g

$$g = \left\lceil \frac{D - len(p)}{2 * (len(k) + len(p))} \right\rceil$$

### 1.3 h

$$\left\lfloor \log_{(2*g+1)} NL \right\rfloor \leq h \leq \left\lceil \log_{(g+1)} \frac{NL}{2} \right\rceil$$

### 1.4 NP

$$NP = \left\lfloor \frac{NR * len(t)}{D * u} \right\rfloor$$

### 1.5 Costo del nested loop

In cui ho R relazione esterna e S relazione interna

- senza predicato di selezione

$$NP_R + NR_R * NP_S$$

- con predicato di selezione

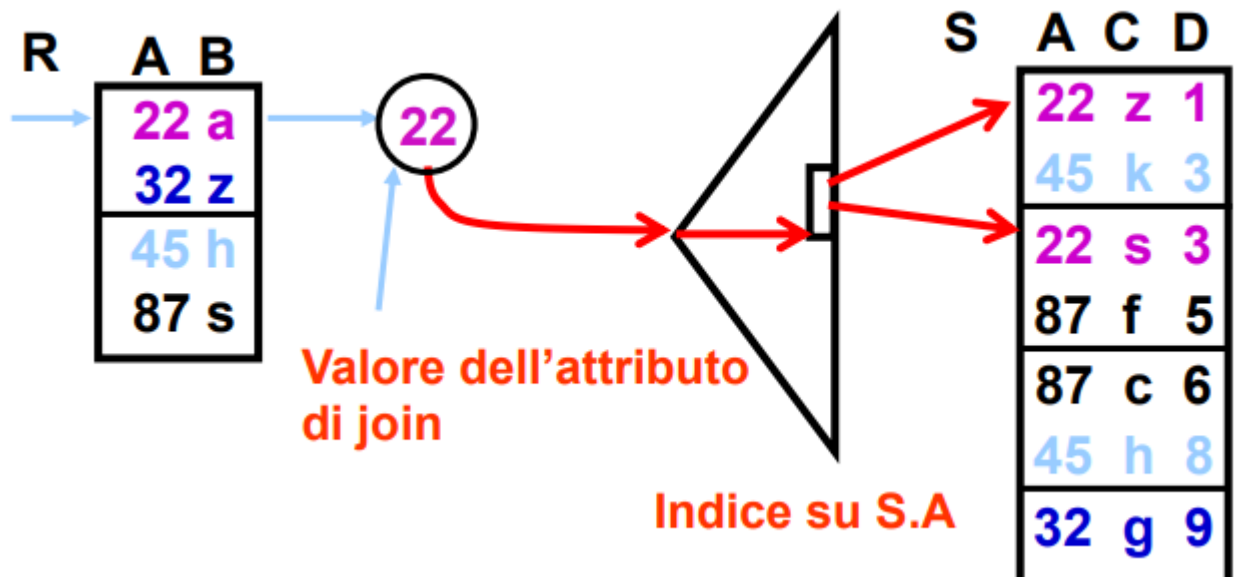
$$NP_R + (sel(pred) * NR_R) * costo(S)$$

## 1.6 Indice clustered

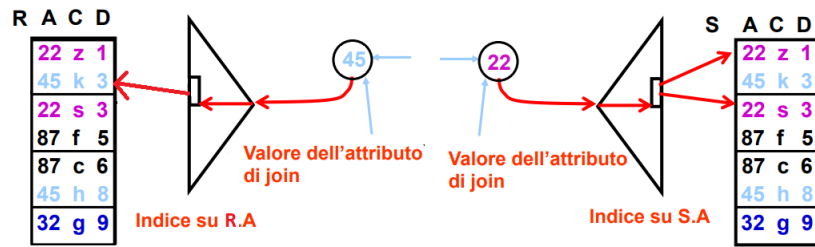
$$Costo(S) = (h - 1) + \left\lfloor \frac{1}{NK} * NL \right\rfloor + \lfloor sel(pred) * NP \rfloor$$

## 1.7 Indice unclustered

$$Costo(S) = (h - 1) + \left\lfloor \frac{1}{NK} * NL \right\rfloor + 1 + \phi\left(\frac{NR}{NK}, NP\right)$$









# MongoDb

## 1 Caratteristiche

MongoDb e' uno dei piu' diffusi database non relazionali orientato ai documenti, di tipo NoSQL, utilizza documenti in un formato JSON al posto delle tipiche tabelle dei sistemi relazionali. Piu' precisamente MongoDB utilizza i BSON ossia JSON binari per rappresentare strutture dati semplici e array associativi (oggetti in MongoDB), un BSON contiene una lista ordinata di elementi appartenenti ai seguenti tipi:

- stringhe
- interi (32 o 64 bit)
- double (numeri a virgola mobile a 64 bit, standard IEEE 754)
- date (numeri interi in millisecondi dal'epoca Unix come riferimento, 1<sup>o</sup> gennaio 1970)
- byte array (dati binari)
- booleani (true e false)
- NULL
- oggetto BSON
- array BSON
- espressioni regolari
- codice JavaScript

## 2 Organizzazione dei dati

### 2.1 Creazione collezioni

Per poter originare i set di dati sui quali lavoriamo siamo partiti da 2 enormi file JSON generati al paragrafo ?????????? //TODO riguarda qua // contenenti POST (A) e COMMENTI(B). Il paramentro \$out posto dopo la query ha creato una collezione a tuttli gli effetti sulla quale poter eseguire le query all'interno della rete del container.

## Referencing

Il referencing di A in B si ottiene facilmente proiettando gli attributi della collezione B, dato che contiene al suo interno la foreign Key di A.

### Referencing di A in B

```
db.B.aggregate([
  {
    $project: {
      "_id" : "$BK",
      "BK" : "$BK",
      "AK" : "$FAK",
      "B1" : "$B1",
      "B2" : "$B2",
      "B3" : "$B3",
      "B4" : "$B4",
      "B5" : "$B5",
      "B6" : "$B6",
      "B7" : "$B7"
    }
  }, {
    $out : "referencing_A_in_B"
  }
])
```

#### Referencing di A in B

```
"_id": 21748,
"BK": 21748,
"AK": 70394,
"B1": 85337,
"B2": 255,
"B3": 6,
"B4": 85337,
"B5": 255,
"B6": 6,
"B7": "Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Fusce lacinia eget arcu et maximus. Ut tempus est sit amet
tortor commodo, sit amet facilisis mi rhoncus. Donec et elit
venenatis, consequat tellus eu, tristique orci. Duis
tristique sem ut nulla ullamcorper, a porta risus efficitur."
```

Cras sed neque et nisl tincidunt vestibulum. Phasellus tristique tempor facilisis. Sed facilisis lectus eros, sed aliquet lacus elementum sed. Integer vel dictum mi. Maecenas pharetra tempus eros, efficitur mattis erat cursus in. Nulla sit amet quam velit. Nullam tempus dictum lacus id porttitor. Vestibulum facilisis pulvinar fermentum. Ut elementum maximus feugiat. In at mollis leo, eu facilisis magna. Vestibulum sed nisi ultricies, tincidunt enim ac, fringilla ex. Phasellus pharetra mollis nisi a fermentum. In nec faucibus nulla, eget molestie magna. Vivamus in gravida ex. Aenean scelerisque gravida ipsum, nec congue enim posuere sit amet. Donec vitae felis id sem congue blandit eget non justo quis."

#### Referencing di A in B

Il referencing di B in A invece richiede un join, un' operazione piu' onerosa per la quale ho dovuto passare il parametro allowDiskUse=true per poter utilizzare a pieno la memoria della macchina e quindi eseguire il join trovando di fatto le chiavi e aggiungendole all'array B che arrivera' a contenere 10 chivi BK per ogni AK dato che il rapporto e' di 1:10.

#### Referencing di B in A

```
db.B.aggregate(  
[  
  {  
    $group: {  
      _id: {"AK" : "$FAK"}, "BK": {$addToSet : "$BK"}  
    }  
  },  
  {  
    $lookup: {  
      from: 'Ap',  
      localField: '_id.AK',  
      foreignField: 'AK',  
      as: 'AK'  
    }  
  },  
  {  
    $project : {"_id" : 0,"BK.FAK" : 0}  
  },  
  {
```

```

$unwind: {
  path: "$AK",
}}, {
  $project : {
    "_id" : "$AK.AK",
    "AK" : "$AK.AK",
    "A1" : "$AK.A1",
    "A2" : "$AK.A2",
    "A3" : "$AK.A3",
    "A4" : "$AK.A4",
    "A5" : "$AK.A5",
    "A6" : "$AK.A6",
    "A7" : "$AK.A7",
    "B" : "$BK"}
  }, {
    $out : "referencing_B_in_A"
  }
], {allowDiskUse:true}
)

```

Referencing di B in A

```

"_id": 8518,
"AK": 8518,
"A1": 6773,
"A2": 253,
"A3": 4,
"A4": 6773,
"A5": 253,
"A6": 4,
"A7": "Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Fusce lacinia eget arcu et maximus. Ut tempus est sit amet
tortor commodo, sit amet facilisis mi rhoncus. Donec et elit
venenatis, consequat tellus eu, tristique orci. Duis
tristique sem ut nulla ullamcorper, a porta risus efficitur.
Cras sed neque et nisl tincidunt vestibulum. Phasellus
tristique tempor facilisis. Sed facilisis lectus eros, sed
aliquet lacus elementum sed. Integer vel dictum mi.
Maecenas pharetra tempus eros, efficitur mattis erat cursus
in. Nulla sit amet quam velit. Nullam tempus dictum lacus
id porttitor. Vestibulum facilisis pulvinar fermentum.

```

```

    Ut elementum maximus feugiat. In at mollis leo, eu
    facilisis magna. Vestibulum sed nisi ultricies, tincidunt
    enim ac, fringilla ex. Phasellus pharetra mollis nisi
    a fermentum. In nec faucibus nulla, eget molestie magna.
    Vivamus in gravida ex. Aenean scelerisque gravida
    ipsum, nec congue enim posuere sit amet. Donec vitae felis
    id sem congue blandit eget non justo quis.",
    "B": [
      182143,
      450648,
      120258,
      151670,
      314940,
      281717,
      335311,
      802016,
      313524,
      537
    ]

```

Referencing di B in A

## Embedding

Per l'embedding di documenti ho dovuto sempre eseguire dei join ma e' stato fondamentale poter aggiungere un intero documento all'interno di un'array tramite il parametro `$$ROOT` che ha permesso un operazione molto piu'veloce e compatta.

## Embedding di A in B

```

db.B.aggregate(
  [
    {
      $lookup: {
        from: 'A',
        localField: 'FAK',
        foreignField: 'AK',
        as: 'A'
      }
    }
  ],{

```

```

    $project: {
      "FAK" : 0,
      "_id" : 0,
      "A._id" : 0
    }
  }
  ,{
    $project : {
      "_id" : "$BK",
      "B1" : 1,
      "B2" : 1,
      "B3" : 1,
      "B4" : 1,
      "B5" : 1,
      "B6" : 1,
      "B7" : 1,
      "A" : 1
    }
  }
  ,{
    $out : "embedding_A_in_B"
  }
]
)

```

\_\_\_\_\_ Embedding di B in A \_\_\_\_\_

```

"_id": 963593,
"B1": 77221,
"B2": 86,
"B3": 1,
"B4": 77221,
"B5": 86,
"B6": 1,
"B7": [ ..... ],
"A": [
  {
    "AK": 70394,
    "A1": 7960,
    "A2": 269,
    "A3": 8,

```



```

    "A4": 7960,
    "A5": 269,
    "A6": 8,
    "A7": [ ..... ]
  }
]

```

Embedding di B in A

## Embedding di B in A

```

db.B.aggregate(
[
  {
    $group: {
      _id: {"AK" : "$FAK"}, "BK": {$addToSet : "$$ROOT"}
    }
  },
  {
    $lookup: {
      from: 'A',
      localField: '_id.AK',
      foreignField: 'AK',
      as: 'AK'
    }
  },
  {
    $project : {"_id" : 0,"BK.FAK" : 0}
  },
  {
    $unwind: {
      path: "$AK",
    }
  },
  {
    $project : {
      "_id" : "$AK.AK",
      "AK" : "$AK.AK",
      "A1" : "$AK.A1",
      "A2" : "$AK.A2",
      "A3" : "$AK.A3",
      "A4" : "$AK.A4",
      "A5" : "$AK.A5",
      "A6" : "$AK.A6",

```

```

        "A7" : "$AK.A7",
        "B" : "$BK"}
    },{
        $project : { "A._id" : 0, "B._id" : 0}
    },{
        $out : "embedding_B_in_A"
    }
],{allowDiskUse:true}
)

```

Embedding di B in A

```

"_id": 48,
"AK": 48,
"A1": 691,
"A2": 154,
"A3": 0,
"A4": 691,
"A5": 154,
"A6": 0,
"A7": [ ..... ],
"B": [
    {
        "BK": 844575,
        "B1": 94813,
        "B2": 140,
        "B3": 2,
        "B4": 94813,
        "B5": 140,
        "B6": 2,
        "B7": [ ..... ]
    },
    {
        "BK": 115006,
        "B1": 36,
        "B2": 483,
        "B3": 8,
        "B4": 36,
        "B5": 483,
        "B6": 8,
        "B7": [ ..... ]
    }
]

```

```
    },  
    .  
    .  
    .  
    .  
    {  
      "BK": 834226,  
      "B1": 77768,  
      "B2": 707,  
      "B3": 6,  
      "B4": 77768,  
      "B5": 707,  
      "B6": 6,  
      "B7": [ ..... ]  
    }  
  ]
```

\_\_\_\_\_ Embedding di B in A \_\_\_\_\_

## 2.2 Indici

Gli indici per gli attributi sono stati creati attraverso uno script in JS che viene eseguito durante la creazione del container.

MongoDB utilizza, se non esplicitamente indicato, come strutture dati per gli indici dei B-tree. Dato che abbiamo scelto di adoperare l'embedding di interi documenti sono stati indicizzati anche gli attributi dei documenti innestati sui quali saranno eseguite delle query, anche sulle foreign del referencing sono stati costruiti indici.

```

1 db.getCollection('embedding_A_in_B').createIndex({'B4': 1});
2 db.getCollection('embedding_A_in_B').createIndex({'B5': 1});
3 db.getCollection('embedding_A_in_B').createIndex({'B6': 1});
4
5
6 db.getCollection('embedding_A_in_B').createIndex({'A.A4': 1});
7 db.getCollection('embedding_A_in_B').createIndex({'A.A5': 1});
8 db.getCollection('embedding_A_in_B').createIndex({'A.A6': 1});
9
10
11 db.getCollection('embedding_B_in_A').createIndex({'B.B4': 1});
12 db.getCollection('embedding_B_in_A').createIndex({'B.B5': 1});
13 db.getCollection('embedding_B_in_A').createIndex({'B.B6': 1});
14
15 db.getCollection('embedding_B_in_A').createIndex({'A4': 1});
16 db.getCollection('embedding_B_in_A').createIndex({'A5': 1});
17 db.getCollection('embedding_B_in_A').createIndex({'A6': 1});
18
19 db.getCollection('referencing_A_in_B').createIndex({'AK': 1});
20
21 db.getCollection('referencing_A_in_B').createIndex({'B4': 1});
22 db.getCollection('referencing_A_in_B').createIndex({'B5': 1});
23 db.getCollection('referencing_A_in_B').createIndex({'B6': 1});
24
25 db.getCollection('referencing_B_in_A').createIndex({'AK': 1});
26
27 db.getCollection('referencing_B_in_A').createIndex({'A4': 1});
28 db.getCollection('referencing_B_in_A').createIndex({'A5': 1});
29 db.getCollection('referencing_B_in_A').createIndex({'A6': 1});
30
31
32 db.getCollection('embedding_A_in_B').createIndex({'A': 1});
33 db.getCollection('embedding_B_in_A').createIndex({'B': 1});

```

```

1 // indexes
2 db.getCollection('embedding_A_in_B').createIndex({'B4': 1});
3 db.getCollection('embedding_A_in_B').createIndex({'B5': 1});
4 db.getCollection('embedding_A_in_B').createIndex({'B6': 1});
5
6 // nested indexes
7 db.getCollection('embedding_A_in_B').createIndex({'A.A4': 1});
8 db.getCollection('embedding_A_in_B').createIndex({'A.A5': 1});
9 db.getCollection('embedding_A_in_B').createIndex({'A.A6': 1});
10
11 // indexes
12 db.getCollection('embedding_B_in_A').createIndex({'A4': 1});
13 db.getCollection('embedding_B_in_A').createIndex({'A5': 1});
14 db.getCollection('embedding_B_in_A').createIndex({'A6': 1});

```

```

15
16 // nested indexes
17 db.getCollection('embedding_B_in_A').createIndex({'B.B4': 1});
18 db.getCollection('embedding_B_in_A').createIndex({'B.B5': 1});
19 db.getCollection('embedding_B_in_A').createIndex({'B.B6': 1});
20
21 //FK
22 db.getCollection('referencing_A_in_B').createIndex({'AK': 1});
23
24 // indexes
25 db.getCollection('referencing_A_in_B').createIndex({'B4': 1});
26 db.getCollection('referencing_A_in_B').createIndex({'B5': 1});
27 db.getCollection('referencing_A_in_B').createIndex({'B6': 1});
28
29 //FK
30 db.getCollection('referencing_B_in_A').createIndex({'AK': 1});
31
32 // indexes
33 db.getCollection('referencing_B_in_A').createIndex({'A4': 1});
34 db.getCollection('referencing_B_in_A').createIndex({'A5': 1});
35 db.getCollection('referencing_B_in_A').createIndex({'A6': 1});
36
37 // array indexes
38 db.getCollection('embedding_A_in_B').createIndex({'A': 1});
39 db.getCollection('embedding_B_in_A').createIndex({'B': 1});

```

Listing 1: making indexes



# Oracle

## 1 Raccolta dati

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Suspendisse et ex vehicula, interdum mi eu, auctor augue. Suspendisse vel sagittis urna. In vitae ligula ipsum. Vivamus mattis neque efficitur, gravida purus facilisis, rhoncus felis. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Nunc bibendum urna porta quam congue ornare. Fusce eget consequat libero. Donec varius justo vel libero malesuada, sit amet dignissim ex tincidunt. In pharetra vestibulum lacus quis laoreet. Donec vel laoreet ex, sed facilisis massa. Mauris commodo velit est. Maecenas non sem elementum, faucibus velit et, bibendum mi. Integer laoreet ex et eros accumsan, nec dapibus nisi vestibulum. Donec sed nunc quis mi gravida congue non vel tortor. Interdum et malesuada fames ac ante ipsum primis in faucibus.

Nullam non pharetra arcu. Donec eget velit elit. Pellentesque sed tortor sodales neque tristique rutrum. Vivamus et ex dolor. Vestibulum lacinia augue sit amet libero mattis pellentesque. In sed congue ante, quis tempus neque. Vestibulum semper eu sapien et vestibulum.

In ac erat ullamcorper, ultricies dolor sit amet, tempus neque. Pellentesque quam erat, ornare ac justo at, cursus luctus ex. Pellentesque at turpis blandit, elementum ex ac, fringilla nibh. Etiam et tincidunt lacus. Integer mattis mi sit amet faucibus rutrum. Sed at nisi commodo, ultricies purus a, tempus lacus. Mauris accumsan enim nisi, in tempor velit pharetra eu. Sed eu turpis et ante sagittis sodales. Duis a tellus id risus ultricies accumsan. Vestibulum bibendum in sapien sit amet rhoncus. Fusce aliquam, metus vel efficitur pulvinar, nibh lacus ultricies nisl, nec rhoncus elit ligula vitae risus. Mauris bibendum eget erat non rutrum. Curabitur in ligula eget lectus facilisis molestie sed sed neque. Vestibulum eu faucibus augue, a luctus augue. Nam lobortis massa non lorem condimentum vehicula. Aliquam efficitur cursus neque, efficitur placerat libero tincidunt non. Nullam.





# CouchDb

## 1 Caratteristiche

E' un sistema di gestione di basi di dati non relazionali. E' stato creato soprattutto per il mondo Web, progettato per poter gestire sia importanti quantita' di richieste attraverso la rete sia per poter immagazzinare i dati in dispositivi mobili ed essere veloce. Utilizza una HTTP/JSON API per poter gestire le richieste ed inviare i dati, quindi fa ampio utilizzo di POST, PUT, DELETE e cosi via. E' diventato un progetto Apache nel 2008, versione OpenSource di CouchServer.

Per poter eseguire query su CouchDB si utilizza Mango, un linguaggio di interrogazione JSON, molto simile a Mongo ma con alcune differenze, tra le piu' importanti c'e' la mancanza dell'operatore di lookup ossia di join, infatti i tempi che presentero' successivamente sulle query di join sono sostanzialmente la somma di query di selezione sulle varie collezioni.

## 2 Popolamento

Per poter popolare i database contenti i diversi tipi di collezione ho utilizzato le collezioni di documenti precedentemente create attraverso MongoDB e per ogni riga, ossia documento, ho effettuato una post all'indirizzo locale del server di CouchDB.

CouchDB richiede che il campo `_id` sia una stringa, quindi ho utilizzato il comando `sed` tramite uno script in bash per poter riutilizzare le collezioni esportate da MongoDB, dato che non potevo utilizzare il join per generare nuovamente le collezioni.

```
t dataJSON/embedding_A_in_B.json | sed -r 's/"_id":([0-9]+)/ "_id": "\1" /' >> dataC
t dataJSON/embedding_B_in_A.json | sed -r 's/"_id":([0-9]+)/ "_id": "\1" /' >> dataC
t dataJSON/referencing_A_in_B.json | sed -r 's/"_id":([0-9]+)/ "_id": "\1" /' >> dataC
t dataJSON/referencing_B_in_A.json | sed -r 's/"_id":([0-9]+)/ "_id": "\1" /' >> dataC
```

```
port_requests
port_json
```

```
l="_http://admin:admin@127.0.0.1:5984/b"
```

```

payload=json.dumps({
    "_id":_id"738878",
    "BK":_id738878,
    "B1":_id94207,
    "B2":_id176,
    "B3":_id3,
    "B4":_id94207,
    "B5":_id176,
    "B6":_id3,
    "B7":_id"Lorem ipsum dolor sit amet, consectetur adipiscing elit....."

    headers={
        'Accept':_id'application/json',
        'Content-Type':_id'application/json'

    response=requests.request("POST",_idurl,_idheaders=headers,_iddata=payload)

    int(response.text)

```

### 3 Indici

Gli indici all'interno di ogni collezione sono stati creati tramite uno script in python con un procedimento simile, per poter ottenere collezioni uguali in tutto e per tutto a quelle di MongoDB. Mango e' un linguaggio di query dichiarativo JSON per CouchDB. Gli indici di Mango con tipo json, sono costruiti usando viste MapReduce.

```

port_requests
port_json
port_os

```

```

collections_a=[_id'referencing_b_in_a',_id'embedding_b_in_a']
collections_b=[_id'embedding_a_in_b',_id'referencing_a_in_b']

```

```

d_a=[_id"A4",_id"A5",_id"A6"]
d_b=[_id"B4",_id"B5",_id"B6"]

```

A

```

l=_id"http://admin:admin@127.0.0.1:5984/a/_index?partitioned=true"

```

```

r_ind_in_ind_a:
payload=json.dumps({
    "index":{
        "fields":[
            ind
        ]
    },
    "name":ind+"-index",
    "type":"json"
})
headers={
    'Content-Type':'application/json'
}

```

```

response=requests.request("POST",url,headers=headers,data=payload)

```

```

print(response.text)
yload=json.dumps({
    "index":{
        "fields":[
            "AK"
        ]
    },
    "name":"AK"+"-index",
    "type":"json"
}

```

```

aders={
    'Content-Type':'application/json'
}

```

```

sponse=requests.request("POST",url,headers=headers,data=payload)
int(response)

```

B

```

l="http://admin:admin@127.0.0.1:5984/b/_index?partitioned=true"
r_ind_in_ind_b:
payload=json.dumps({
    "index":{
        "fields":[
            ind
        ]
    }
}

```

```

        ]
    },
    "name": ind+"-index",
    "type": "json"
})

headers={
    'Content-Type': 'application/json'
}

response=requests.request("POST",url,headers=headers,data=payload)

print(response.text)
payload=json.dumps({
    "index":{
        "fields":[
            "BK"
        ]
    },
    "name": "BK"+"-index",
    "type": "json"

})

headers={
    'Content-Type': 'application/json'

}

response=requests.request("POST",url,headers=headers,data=payload)
int(response)

r_collection_in_collections_a:
url="http://admin:admin@127.0.0.1:5984/"+collection+"/_index?partitioned=true"
for ind_in_ind_a:
    payload=json.dumps({
        "index":{
            "fields":[
                ind
            ]
        },
        "name": ind+"-index",
        "type": "json"
    })

```

```

headers={
    'Content-Type': 'application/json'
}

response=requests.request("POST",url,headers=headers,data=payload)

print(response.text)

r_collection_in_collections_b:
url="http://admin:admin@127.0.0.1:5984/"+collection+"/_index?partitioned=true"
for_ind_in_ind_b:
    payload=json.dumps({
        "index":{
            "fields":[
                ind
            ]
        },
        "name":ind+"-index",
        "type":"json"
    })
    headers={
        'Content-Type': 'application/json'
    }

    response=requests.request("POST",url,headers=headers,data=payload)

    print(response.text)

l="http://admin:admin@127.0.0.1:5984/referencing_a_in_b/_index?partitioned=true"
payload=json.dumps({
    "index":{
        "fields":[
            "AK"
        ]
    },
    "name": "AK"+"-index",
    "type":"json"

    aders={

```

```
'Content-Type':_ 'application/json'
```

```
sponse=_requests.request("POST",_url,_headers=headers,_data=payload)
```

```
emb_AB
```

```
l=_ "http://admin:admin@127.0.0.1:5984/embedding_a_in_b/_index?partitioned=true"
```

```
r_ind_in_ind_a:
```

```
payload=_json.dumps({
```

```
  _ "index":_{
```

```
    _ _ _ "fields":_[
```

```
      _ _ _ _ "A."+_ind
```

```
    _ _ _ _]
```

```
  _ _ },
```

```
  _ _ "name":_ "A."+_ind+_ "-index",
```

```
  _ _ "type":_ "json"
```

```
})
```

```
headers=__{
```

```
  _ _ 'Content-Type':_ 'application/json'
```

```
}
```

```
response=_requests.request("POST",_url,_headers=headers,_data=payload)
```

```
print(response.text)
```

```
emb_BA
```

```
l=_ "http://admin:admin@127.0.0.1:5984/embedding_b_in_a/_index?partitioned=true"
```

```
r_ind_in_ind_b:
```

```
payload=_json.dumps({
```

```
  _ _ "index":_{
```

```
    _ _ _ "fields":_[
```

```
      _ _ _ _ "B."+_ind
```

```
    _ _ _ _]
```

```
  _ _ },
```

```
  _ _ "name":_ "B."+_ind+_ "-index",
```

```
  _ _ "type":_ "json"
```

```
})
```

```
headers=__{
```

```
  _ _ 'Content-Type':_ 'application/json'
```

```
}
```

```
response=_requests.request("POST",_url,_headers=headers,_data=payload)
```

```
print(response.text)
```

```
ref_BA
```

```
l="_"http://admin:admin@127.0.0.1:5984/referencing_b_in_a/_index?partitioned=true"
```

```
yload=_json.dumps({
```

```
"index":_{
```

```
  _"fields":_[
```

```
    _"B"
```

```
  _]
```

```
},
```

```
"name":_"B"+_"-index",
```

```
"type":_"json"
```

```
aders=_{
```

```
'Content-Type':_'application/json'
```

```
sponse=_requests.request("POST",_url,_headers=headers,_data=payload)
```

```
int(response.text)
```





Minimo documento

Appendice



Ringraziamenti