Relazione progetto 2 Cfu
Complementi di basi di dati

Ottimizzazione delle interrogazioni in MongoDb

Introduzione

Il database utilizzato è LiquorLicenses importato come file .csv; per importarlo anche nel proprio ambiente, le azioni da eseguire sono le seguenti:

1. Scaricare il dataset dal sito https://app.enigma.io/;
2. Aprire il prompt e spostarsi nella cartella in cui è stato salvato il file .cvs;
3. Eseguire il comando *mongoimport* in questo modo:
   mongoimport -d test -c LiquorLicenses --drop --type csv --file liquor-licenses.csv --headerline

Al termine di queste operazioni sarà possibile trovare una collection 'LiquorLicenses' nel db 'test' con poco più di 2 milioni di record; i documenti al suo interno sono tra loro omogenei.
Lanciando il comando *db.LiquorLicenses.stats()* possiamo osservare lo stato iniziale della collection, come per esempio:

- Numero di documenti: 2.010.471,
- Dimensione totale: 1.738.647.823.0 bytes,
- Dimensione media di un documento: 864 bytes,
- Numero di indici: 1 (creato di default).

Prima di continuare, è bene specificare che, per default, MongoDB crea un indice singolo sul campo _id di ogni collezione di tipo *unique*, ossia un indice dove può esistere un solo documento con un dato valore, infatti ogni documento deve avere un ObjectId univoco.

Interrogazioni senza strutture d'accesso

Come ho scritto nell'introduzione, MongoDb definisce di default un indice univoco sul campo _id, dunque un'interrogazione su tale campo richiederebbe un tempo molto basso, dovuta alla presenza di tale indice. Notiamo subito con un'interrogazione semplice

db.LiquorLicenses.find();

che i documenti sono ordinati in base al campo license_state.
Useremo la funzione *explain("executionStats");* per sapere il piano di esecuzione delle nostre interrogazioni e quindi informazioni come numero di documenti tornati, numero di documenti scansionati, tempo impiegato ecc.

1. *db.LiquorLicenses.find({license_state: "Texas"}).explain("executionStats");*

Tutte le licenza che hanno *license_state* = *Texas*.

- tempo stimato: 1.560 msec
- tempo reale: 1.550 msec
- record tornati: 60.893
- record esaminati: tutti.

2. *db.LiquorLicenses.find({license_state: "Texas", license_expire_date: {$regex: /2016/i} },*

{_id: 0, license_number: 1, license_address_city: 1}

).explain("executionStats");

Torna *license_number* e *license_address_city* delle licenze che hanno *license_state* = Texas e *license_expire_date* nel 2016

- tempo stimato: 1.740 msec
- tempo reale: 1.795 msec
- record tornati: 28.005

- record esaminati: tutti.

3. *db.LiquorLicenses.find({license_state: "Texas", license_expire_date: "2016-09-11"}).explain("executionStats");*

Tutte le licenze che hanno *license_state* = Texas e che hanno *license_expire_date* il 2016-09-11.

- tempo stimato: 6.660 msec
- tempo reale: 6.653 msec
- record tornati: 36
- record esaminati: tutti.

4. *db.LiquorLicenses.find({"license_expire_date": {$regex: /2020/i}}).explain("executionStats");*

Tutte le licenze che hanno *license_expire_date* nel 2020.

- tempo stimato: 2.440 msec
- tempo reale: 2.690 msec
- record tornati: 427
- record esaminati: tutti

5. *db.LiquorLicenses.find({"license_expire_date":*
{$regex: /2020/i}},
{"license_number": 1, "license_state": 1, _id: 0, "license_expire_date": 1}
).explain("executionStats");

Torna il *license_number, license_state* delle licenze che hanno *license_expire_date* nel 2020.

- tempo stimato: 2.400 msec
- tempo reale: 2.640 msec
- record tornati: 427
- record esaminati: tutti.

6. *db.LiquorLicenses.find({"email": {$exists: true, $ne: ""}}).explain("executionStats");*

Tutte le licenze che hanno *email* non nulla e non vuota.

- tempo stimato: 4870 msec
- tempo reale: 4920 msec
- record tornati: 42.479
- record esaminati: tutti.

7. db.LiquorLicenses.find({"email": "STEVELANDER@BAMALANES.COM", "licensee_principal" : "BAMA LANES INC"});

Tutte le licenze che hanno campo *email* = STEVELANDER@BAMALANES.COM e *license_principal* = BAMA LANES INC

- tempo stimato: 3590 msec
- tempo reale: 3723 msec
- record tornati: 4
- record esaminati: tutti.

Interrogazioni con uso della function *limit()*

Il modo più rapido per migliorare il tempo di esecuzione delle interrogazioni è inserire, tramite la funzione *limit()* prima dell'*explain()*, un numero di documenti massivo da tornare come risultato dell'interrogazione.
Ho rifatto tutte le interrogazioni precedenti aggiungendo limit(5), e ho ottenuto i seguenti risultati:
N.B.: da ora in poi userò i numeri 1,2,…..,7 per riferirmi alle interrogazioni precedenti specificando eventuali modifiche o aggiunte.

Query 1:

- tempo stimato: 1.330 msec
- tempo reale: 1.473 msec
- record tornati: 5
- record esaminati: 1.835.429

Query 2:

- tempo stimato: 1.420 msec
- tempo reale: 1.450 msec
- record tornati: 5
- record esaminati: 1.835.429.

Query 3:

- tempo stimato: 6.460 msec
- tempo reale: 6.623 msec
- record tornati: 5
- record esaminati: 1.838.287.

Query 4:

- tempo stimato: 1.620 msec
- tempo reale: 1.757 msec
- record tornati: 5
- record esaminati: 1.788.068

Query 5:

- tempo stimato: 1.560 msec
- tempo reale: 1.705 msec
- record tornati: 5
- record esaminati: 1.788.068

Query 6:

- tempo stimato: 10 msec
- tempo reale: 6 msec
- record tornati: 5
- record esaminati: 1.897.

Query 7:

- tempo stimato: 3570 msec
- tempo reale: 3762 msec
- record tornati: 4
- record esaminati: tutti.

Per quanto riguarda le prime tre query (quelle che controllano che le licenze siano del 'Texas'), abbiamo un lieve miglioramento dovuto al fatto che vengono analizzati meno documenti rispetto al caso precedente (200.000 in meno); essendo la collection ordinata in base alla *license_state* è comunque necessario analizzare quasi tutti i documenti, infatti 'Texas' si trova in fondo in base a tale ordine.

Anche per la 4 e la 5 abbiamo un lieve miglioramento dovuto ancora alla minor quantità di documenti considerati.

Nella Query 6 invece abbiamo un netto miglioramento; è sufficiente analizzare 1.897 per trovarne 5 che abbiamo il campo *email* valido.

Nella Query 7 non abbiamo nessun miglioramento: specificare o meno limit(5) è indifferente, perché all'interno della collection ce ne sono solo 4 che verificano le sue condizioni e quindi comunque tutti i documenti verranno considerati.

Interrogazioni con uso di un indice su campo singolo

Ho aggiunto alla collection i seguenti indici:

db.LiquorLicenses.ensureIndex({ license_state: 1}, {sparse: true});
tempo di creazione: 9.367 msec; indiece su *license_state*.

db.LiquorLicenses.ensureIndex({ license_expire_date: 1});
temposdddfcqer di creazione: 12.728 msec; indice su *license_expire_date*.

db.LiquorLicenses.ensureIndex({ email: 1});
tempo di creazione: 12.425 msec; indice su *email*.

L'opzione *sparse* serve per indicare che l'indice non deve contenere riferimenti ai documenti che non hanno il campo indicizzato; è più leggero rispetto agli altri indici, ma richiede più tempo di esecuzione in alcune situazioni.

Se il campo è specificato con valore 1 allora l'indice sul campo seguirà un ordine crescente, se è specificato –1 seguirà un ordine decrescente.

Quando nelle condizioni della query è presente: *$exist:true, $gt, $gte, $lt, $lte, $type* ossia delle *partialFilterExpression* è utile creare un indice che non specifichi né *sparse* né *unique* per evitare comportamenti particolari (come aumento del tempo di esecuzione).

Per utilizzare gli indici sfrutteremo la funzione *hint()*, in quanto MongoDB effettua delle assunzioni sui dati per effettuare le query, non sempre corrette; tramite *explain()* siamo in grado di vedere tali assunzioni e vedere se ha scelto un piano di esecuzione favorevole. Per esempio se, dovendo scegliere tra due indici, avesse scelto quello più svantaggioso nel caso della query da eseguire, è possibile suggerire a MongoDB l'indice da usare. La funzione *hint()* è da aggiungere prima della chiamata alla funzione *explain()*, ma dopo la *find()* (esattamente come la funzione *limit()*)

Per le query 1,2 e 3 si deve aggiungere:
.hint({license_state: 1})
Per le query 4 e 5 si deve aggiungere:
.hint({ license_expire_date: 1 })

Per le query 6 e 7 si deve aggiungere:
.hint({ email: 1 })

Eseguendo nuovamente le interrogazioni otteniamo:

Query 1:

- tempo stimato: 110 msec
- tempo reale: 201 msec
- record tornati: 60.893
- record esaminati: 60.894.

Query 2:

- tempo stimato: 334 msec
- tempo reale: 316 msec
- record tornati: 28.005
- record esaminati: 60.893.

Query 3:

- tempo stimato: 172 msec
- tempo reale: 180 msec
- record tornati: 36
- record esaminati: 60.893

Query 4:

- tempo stimato: 2.440 msec
- tempo reale: 2.642 msec
- record tornati: 427
- record esaminati: 427.

Query 5:

- tempo stimato: 2.800 msec
- tempo reale: 2.843 msec
- record tornati: 427
- record esaminati: 427.

Query 6:

- tempo stimato: 70 msec
- tempo reale: 184 msec
- record tornati: 42.481
- record esaminati: 42.481.

Query 7:

- tempo stimato: 0 msec
- tempo reale: 1 msec
- record tornati: 4
- record esaminati: 4.

Nelle query 4 e 5, si ha un aumento del tempo rispetto all'utilizzo di limit(), ma lo stesso tempo rispetto alla prima interrogazione. Questo perché per effettuare il controllo sulla condizione specificata nella query è comunque necessario cerare un numero di 'work units' pari al numero di documenti presenti nella collection, rendendo cosi ininfluente l'utilizzo dell'indice. Anche se i documenti analizzati sono 427, MongoDb esamina tutte le chiavi per poter valutare la condizione di *regex*.

Nelle query 6 e 7, il tempo è migliorato rispetto a entrambi i casi precedenti e anche il numero di documenti analizzati è diminuito. In particolare nella 7 siamo riusciti a ottenere un risultato ottimo: 4 documenti analizzati, 4 tornati e tempo di esecuzione stimato 0.

Nelle query 1,2 e 3 sia ha un netto miglioramento: il tempo resta comunque al di sopra dei 100 msec.

Interrogazioni con uso di un indice su due campi

Aggiungo nuovi indici cosi definiti:

db.LiquorLicenses.ensureIndex({ license_state:1, license_expire_date: 1}, {sparse: true});
Tempo di creazione: 15.738 msec.

db.LiquorLicenses.ensureIndex({ email: 1, license_principal: 1});
Tempo di creazione: 12.940 msec.

Per le query 1, 4 , 5, 6 non ho creato nessun indice su due campi in quanto le condizioni si basano solo sull'attributo email e non essendoci nessun controllo su altri campi questa ulteriore casistica sarebbe inutile.

Per le altre, ho modificato il vecchio hint( ) con:
Per le query 2 e 3:
.hint({ license_state:1, license_expire_date: 1 });
Per le query 7:
.hint({ email: 1, license_principal: 1 });

Ho ottenuto i seguenti risultati:

Query 2:

- tempo stimato: 260 msec
- tempo reale: 260 msec
- record tornati: 28.005
- record esaminati: 28.005.

Query 3:

- tempo stimato: 0 msec
- tempo reale: 1 msec
- record tornati: 36
- record esaminati: 36

Query 7:

- tempo stimato: 0 msec
- tempo reale: 1 msec

- record tornati: 4
- record esaminati: 4.

Anche nella query 3 siamo riusciti a ottimizzare il tempo nei migliori dei modi; nella query 2 il miglioramento è lieve rispetto all'indice precedente, questo perché è presente una condizione che si affida all'utilizzo di $regex; nella query 7 non c'era niente da migliorare, ma comunque l'esecuzione è rimasta ottima.

Nel caso di controlli con $regex e un indice sul campo considerato, MongoDb valuta l'espressione regolare sui valori dell'indice; questo potrebbe portare dei miglioramenti; nel caso della query 2 il numero di documenti esaminati è diminuito, ma il numero di chiavi esaminate è rimasto lo stesso del caso precedente (60.893).

Le query che non siamo ancora riusciti a migliorare sono dunque la 4 e la 5. Avendo specificato "2020" MongoDb farà una scansione completa delle chiavi sull'indice e poi andrà a recuperare i documenti corrispondenti. Se lo modifichiamo con "^2020" analizzerà solo l'intervallo di valori dell'indice che iniziano con 2020. Modificando le condizioni e otteniamo:

Query 4:

- tempo stimato: 0 msec
- tempo reale: 1 msec
- record tornati: 427
- record esaminati: 427.

Query 5:

- tempo stimato: 0 msec
- tempo reale: 10 msec
- record tornati: 427
- record esaminati: 427.

Ovviamente lo stesso ragionamento può essere fatto con la query 2:
2016 -> ^2016

Query 2:

- tempo stimato: 150 msec
- tempo reale: 164 msec
- record tornati: 28.005
- record esaminati: 28.005.

Il tempo per questa query era già basso, ma con questo trucco si è abbassato di altri 100 msec.

Analisi dei tempi

Per vedere l'analisi dei tempi in formato grafico e tabellare, consultare il file AnalisiDeiTempi.pdf