

# MongoDB教育訓練-08

## Time to Live (TTL) Indexes

---

- Alternative way to use TTL Indexes
  - Put the date you want it deleted in a field (expire-on)
  - Add a TTL with expireAfterSeconds set to 0
- Watch out of unplanned write load
  - Better to write your own programmatic data cleaner.
  - Schedule using a framework like cron/scheduler/Atlas

## Native text Indexes

---

- Superseded in Atlas by Lucene - but relevant to on-premise
- Indexes tokens (words, etc.) used in string fields.
  - It allows you to search for 'contains.'
- Algorithm
  - Split text fields into a list of words.
  - Drop language-specific stop words ("the", "an", "a", "and").
  - Apply language-specific suffix stemming
  - "running", "runs", "runner" all become "run".
  - Take the set of stemmed words and make a multikey index.
- MongoDB supports text search for several western languages.
- Queries are OR by default.
- Can be compound indexes

# Native text Indexes - limits

- Logical AND queries can be performed by putting required terms in quotes
  - This can also be used for required phrases "ocean drive"
  - This applied as secondary filter to an OR query for all terms
  - This makes AND and Phrase queries very inefficient.
- No fuzzy matching
- Many index entries per document (slow to update)
- No wildcard searching
- Indexes are smaller than Lucene though

"ocean drive" - Ocean OR Drive

"\"ocean\" \"drive\"" - Ocean AND Drive

"\"ocean drive\"" - Ocean and Drive as two consecutive words with one space

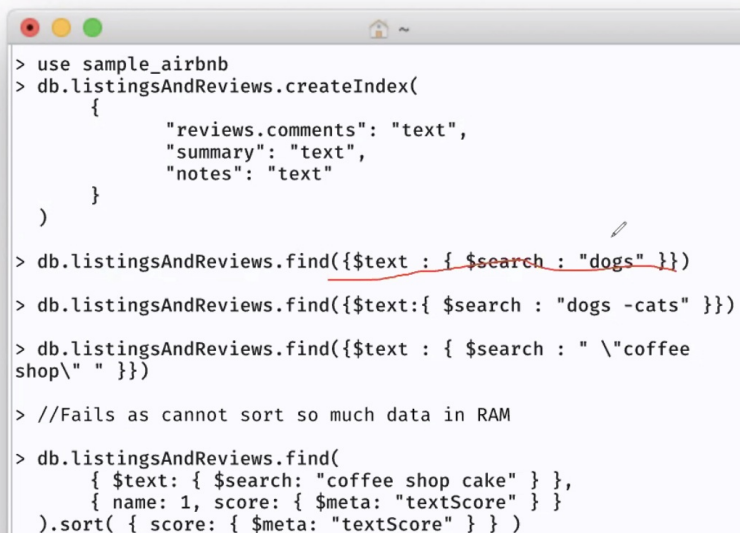
## Text Index Example

Only one text index per collection, so create on multiple fields.

Index all fields with

```
db.collection.createIndex(  
  { "$**": "text" }  
)
```

Use \$meta and \$sort to order if results are small.



```
> use sample_airbnb  
> db.listingsAndReviews.createIndex(  
  {  
    "reviews.comments": "text",  
    "summary": "text",  
    "notes": "text"  
  }  
)  
  
> db.listingsAndReviews.find({$text : { $search : "dogs" }})  
> db.listingsAndReviews.find({$text:{ $search : "dogs -cats" }})  
> db.listingsAndReviews.find({$text : { $search : "\"coffee shop\" " }})  
  
> //Fails as cannot sort so much data in RAM  
  
> db.listingsAndReviews.find(  
  { $text: { $search: "coffee shop cake" } },  
  { name: 1, score: { $meta: "textScore" } }  
)  
.sort( { score: { $meta: "textScore" } } )
```

## Wildcard Indexes

- 可以加入所有欄位，但只能命中一個欄位
- 儲存空間的最大
- 不知道要對什麼做搜索的時候適用

# Wildcard Indexes

---

- Dynamic schema means it's hard to index **all** fields
  - There are alternative schemas that we will see later
- Wildcard indexes index all fields, or a subtree
- Index Entries treat the fieldpath as the first value
  - Normal index on "user.firstname" Index contains "Job","Joe","John" as keys.
  - Wildcard Index on "user" contains the fieldnames in the keys:  
"firstname.Job", "firstname.Joe", "firstname.John",  
and any other other fields in user e.g.  
"lastname.Adams", "lastname.Jones", "lastname.Melville"
  - What does that do to the index size?
  - What are the performance and hardware implications?
  - This feature is easy to overuse/misuse.
  - Indexing correctly matters - not just "index everything"

- 使用hints指定索引
- 支援regular (只有前綴固定 ^開頭支援，範圍查詢)
- 可以設定collation (定序)

## More Index usage tips (or hints)

---

- We can use 'hint' to tell MongoDB what Index to use in find.
- We can use an index for a Regular Expression match,
  - If anchored at the start { name: /^Joh/ }
  - But beware of the case, and also be aware that's a range query >= "Joh" and < "Joi"
- We can set a collation order for Indexes (No diacritics, case insensitive, etc.)
  - Preferred collation can be specified for collection and views.
  - Indexes and Queries can specify what collation use if not the default.

- 增加一個索引增加10%消耗
- 記憶體最好容納最常需要的索引
- \$indexStarts() 可以統計索引的命中率(server 服務以來)
- {a: 1} {a: 1,b:1} 前面這個與後面重複可以刪除

## Even more Index usage tips

---

- Indexes have ~10% overhead on writing per index entry (beware of multikey)
- Accessing Index should ideally be from RAM based cache, if we don't have it cache, it will fetch data from disk.
- We can use `$indexStats()` aggregation to see how much each index is used.
- More Indexes generally means more RAM required.
- Remove indexes that aren't used by any queries to save CPU and RAM.
- If one index is a prefix of another it is redundant { a: 1} and {a:1,b:1}
- Indexes are mostly just like RDBMS indexes - but queries are simpler.

## Indexes in Production

---

To build an index, one has to read whole data, which may be much bigger than the RAM size in the production. So, there were two ways of creating an index in the previous versions: Foreground and Background.

### Foreground Index (MongoDB Pre 4.2)

Foreground index builds were fast but required blocking all read-write access to the parent database of the collection being indexed for the duration of the build.

### Background Index (MongoDB Pre 4.2)

Background index builds were slower and had less efficient results but allowed read-write access to the database and its collections during the build process.

### Hybrid Index (MongoDB 4.2+)

Does not lock the server and builds quickly. It is in the newer version of MongoDB.

## 滾動索引

- 暫停一個結點建立索引，再加入叢集
- 重複兩次後將建好索引主機升為master，下master建索引

## Indexes in Production: Rolling Build

---

- In production, many maintenance tasks are done in a Rolling Manner
  - Rolling updates are fast with minimal impact on production.
  - This can include creating a new index.
- Change performed on temporarily offline secondary
- Then secondary added back to the replica set
  - Secondary catches up using the transaction log.

# Indexes in Production: Hidden Indexes

---

- Hidden indexes allow you to prevent the database using an index without dropping it.
- Creating an Index is expensive and takes time.
- Dropping an Index you no longer think you need could be a risk
  - Hiding it lets you test if it is OK to drop it.
- If you Hide an index.
  - It is no longer used in any operations like find() or update()
  - It is still updated and can be re-enabled at any time.
  - It still applies unique constraints
  - If it is a TTL index then documents will still be removed.

## Recap

---

- Indexes improve efficiency and speed of reads
- Every query should use an index
- Compound indexes are the most used ones in MongoDB
- The order in a compound index is very important.
- MongoDB can explain() how an operation is being indexed.

---

🕒 修訂版本 #3

★ 由 treeman 建立於 17 🕒 2021 21:47:24

✍ 由 treeman 更新於 5 🕒 2023 10:14:59