



#codemash

MongoDB – Zero to Sharding: Part 2.2 : Indexing

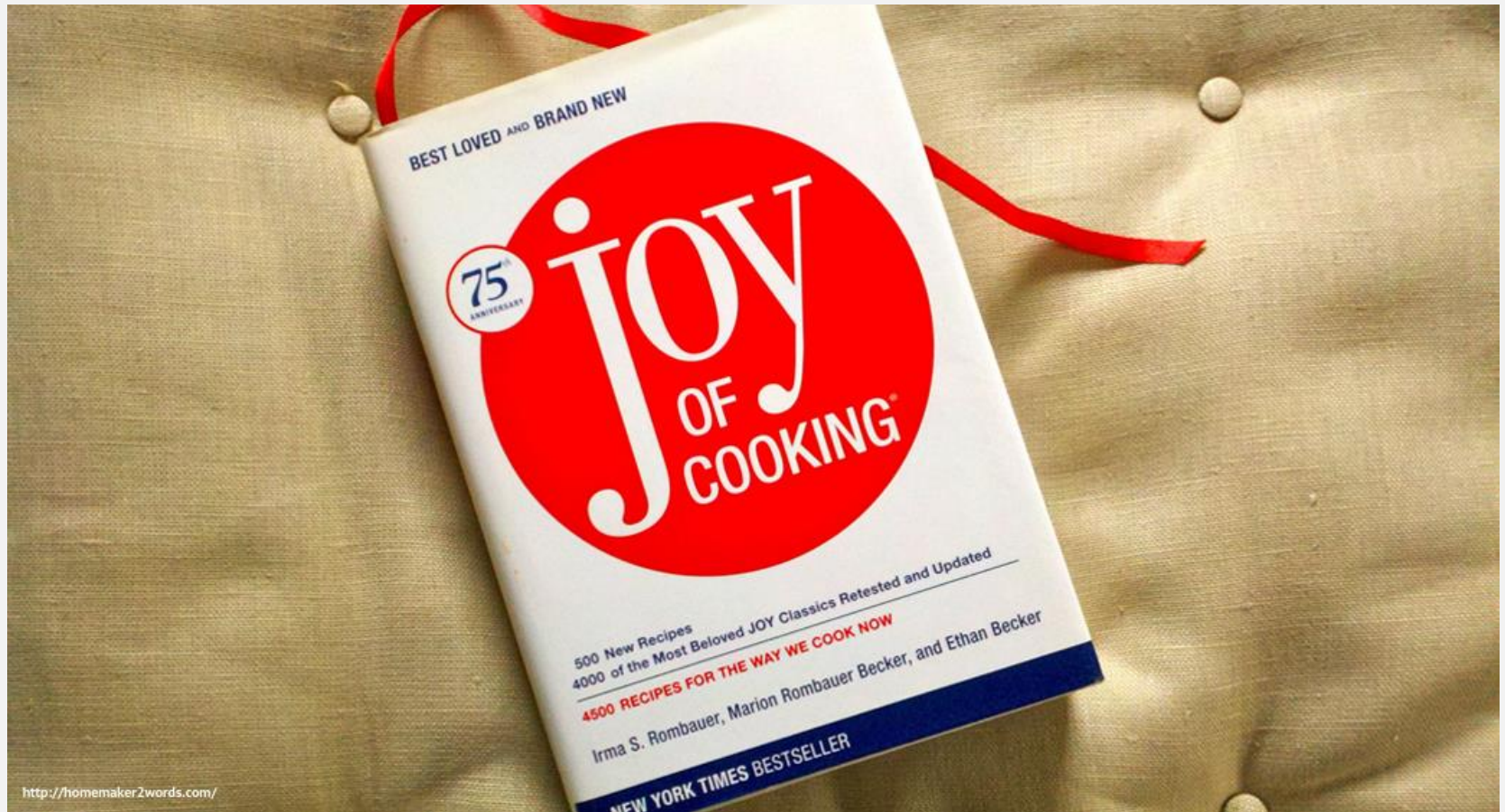
Sridhar Nanjundeswaran

Engineer, MongoDB Inc.

@snanjund

Agenda

- Working with indexes in MongoDB
- Optimize your queries
- Avoiding common mistakes



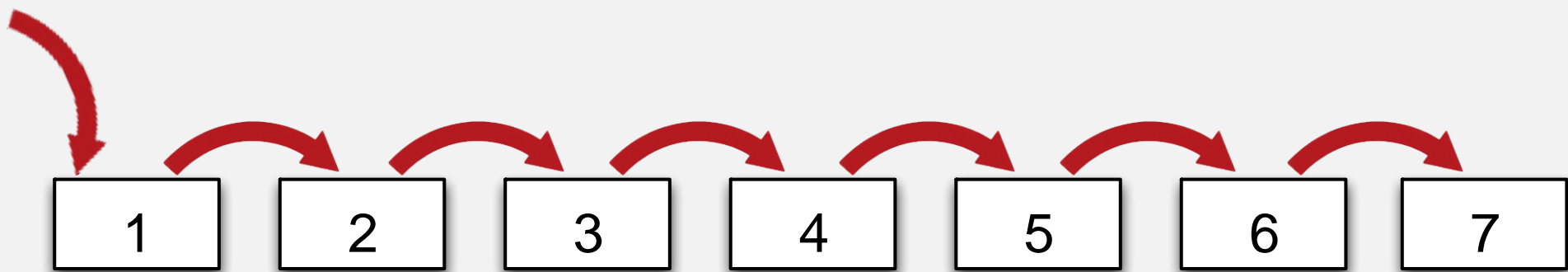
What are indexes?

72	SUPER EASY BURRITOS FOR A CROWD	118	SAUSAGE BISCUITS	139	EASY CHERRY CHEESE CAKE	161	ZUCCHINI CUSTARD PIE	180
	SWEET AND SOUR CHICKEN WITH RICE	118	STICKY BUNS	140	EGG CUSTARD PIE	161	COOKIES & CANDY	
	SWEET CAJUN BEEF STEW	119	SUNFLOWER BREAD	141	EGGNOG PIE	162	AUNT FLORENCE'S CHRISTMAS COOKIES	181
	SWEET N SOUR CHICKEN BREASTS	120	SUPER EASY INDIAN CHAPATIS	141	FLAT APPLE PIE	162	BEST EVER CHOCOLATE CHIP COOKIES	181
	TAMALES	120	SWEET CORNBREAD CAKE	142	FLOURLESS CHOCOLATE CAKE	163	BISCOTTI QUEEN'S ITALIAN LEMON BISCOTTI	182
	THE KING'S QUICHE	122	WHITE BREAD (GUMPBREAD)	142	FOOLPROOF FLAN	163	CHOCOLATE CHIP COOKIE BARS	183
	TOUCAN CHILI	122	WHOLE WHEAT BANANA BREAD	143	FRESH STRAWBERRY CAKE	164	CHOCOLATE CRACKER CRUNCH BARS	183
	WHITE CHICKEN CHILI (CROCK POT RECIPE)	123	ZUCCHINI BREAD	144, 145	GOLDEN SPONGE CAKE	164	CHOCOLATE RIBBONS	183
	ZESTY CHEESY RAVIOLI	124			GRANDMA BURNIE'S BLUEBERRY CRUNCH	165	CREAMY PRALINES	184
	BREADS & ROLLS		DESSERTS		GRANDMA SADIE'S CREAMY CHEESECAKE	165	EASY FUDGE	184
	BANANA CREAM CHEESE BREAD	125	10 EGG POUND CAKE	147	GRANDMA'S EGG CUSTARD	166	FAST PEANUT BUTTER FUDGE	185
	BEER CHEESE BREAD	125	216 AND ¾ PIE CRUST	147	HOMEMADE ICE CREAM	166	FORGET ME COOKIES	185
	BEST BANANA BREAD EVER	125	5 MINUTE CHOCOLATE MUG CAKE	148	LEMON CHESS PIE	166	FRUITY SPRITZ COOKIES	185
	BRAN OVERNIGHT REFRIGERATOR ROLLS	126	AMISH APPLE PIE	148	LOVELY LEMON PIE	167	GRAHAM CRACKER MERINGUE COOKIES	186
	BROCCOLI CORNBREAD	126	ANGEL FOOD CAKE	149	MAMAW'S CHOCOLATE PIE	167	GRAMA PAT'S OATMEAL COOKIES	186
	BUTTER DIPS	127	APPLE CINNAMON CAKE	149	MAW'S PECAN PIE	168	GRANDMA FLO'S SUPER EASY ROCKY ROAD	187
	BUTTER HORN ROLLS	127	APPLE CRISP	150	MOIST LEMON LOVER'S POUND CAKE	168	GRANDMA'S CRUNCHY COOKIES	187
	CARROT COCONUT BREAD	128	APPLE SPICE CAKE WITH CREAM CHEESE FROSTING	150	MOUNTAIN MAMMA MUD PUDDING CAKE	169	HOLIDAY SUGAR COOKIES	187
	CHEESE BISCUITS	128	AUNT SALLY'S BUÑUELOS	151	NANNIES' EGG PIE	170	LACE COOKIES	188
	CHEROKEE FRYBREAD	128	BAKED CUSTARD	151	NO-BAKE CHEESECAKE	170	MICROWAVE CARAMEL POPCORN	188
	CHERRY CORNBREAD	129	BAKLAVA	151	OATMEAL CHOCOLATE BARS	170	MOCHA FROSTED DROPS	189
	CINNAMON SWIRL BREAD	129	BERRY TRIFLE	152	OLD FASHIONED POUND CAKE	171	MOLASSES COOKIES	189
	CRANBERRY BREAD	130	BLACKBERRY COBBLER	153	ORANGE BALLS	171	MOMMA'S SUGAR COOKIES	190
	EASY SOFT RYE BREAD	131	BLUEBERRY COOLWHIP PIE	153	ORANGE CRANBERRY BARS	172	MOM'S SUGAR COOKIES	190
	EGG BREAD	131	BREAD AND BUTTER PUDDING	153	PEACH COBBLER	172	NO BAKE COOKIES	191
	FRENCH TOAST CUSTARD	132	BUTTER POUND CAKE	154	PEACH CRISP	172	NO-BAKE COOKIES	191
	GOLDEN CRESCENT ROLLS	132	BUTTERSCOTCH PIE	154	PEACH PIZZA	173	OATMEAL PEANUT BUTTER CHOCOLATE CHIP	
	GRAMA'S FLOUR TORTILLAS	133	CAKE & ICE CREAM DELIGHT	154	PEANUT BUTTER ICING	173		
	KUCHEN BREAD (GERMAN SWEET BREAD)	133	CARAMEL FLAN	155	PERFECT CHERRY PIE	173		
	LIGHTER THAN AIR PAN ROLLS	134	CHEESECAKE WITH PASTRY	156	PUMPKIN DELIGHT	174		
	http://files.backyardchickens.com/images/BYC-Cookbook-index-3.gif				PUMPKIN DESSERT	174		

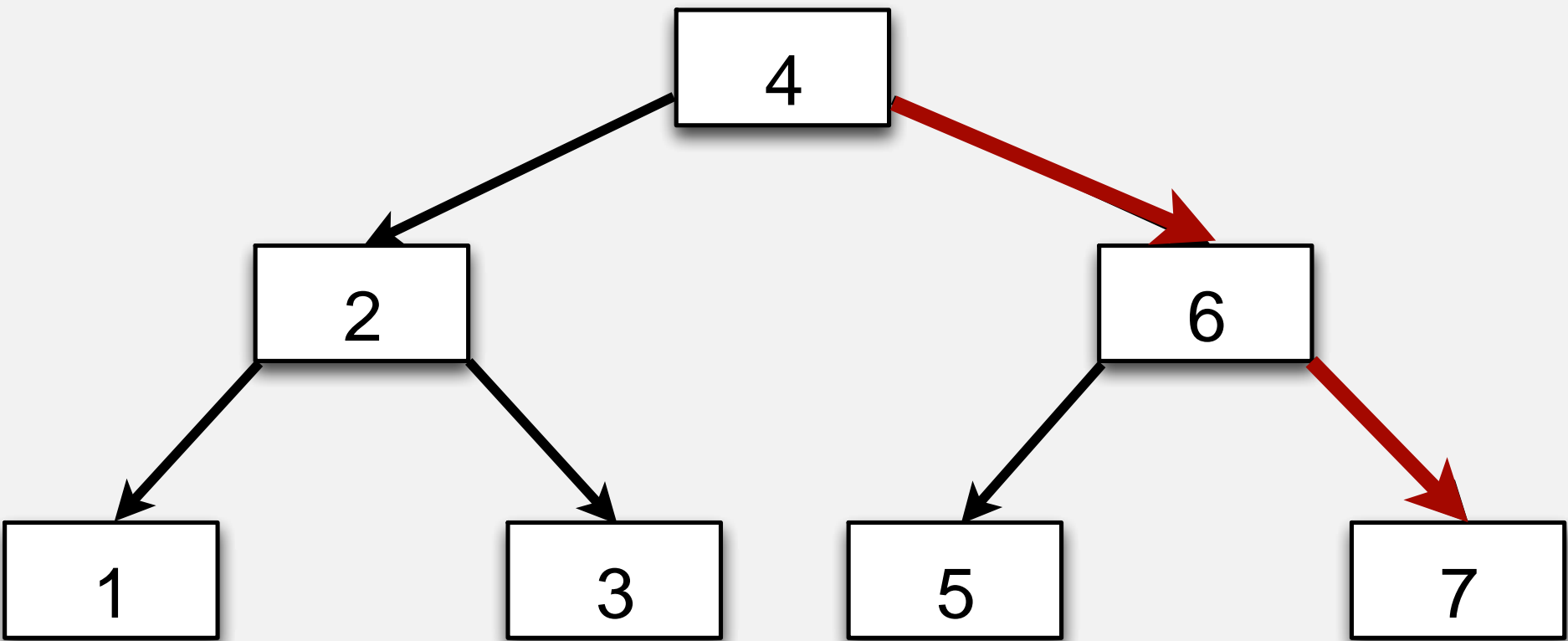
Consult the index!



Linked List

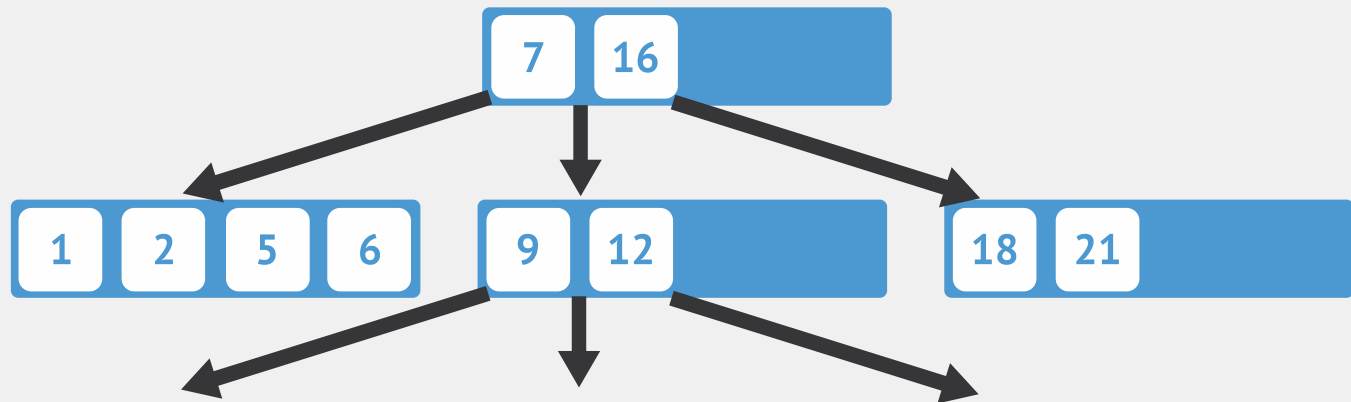


Finding 7 in Linked List



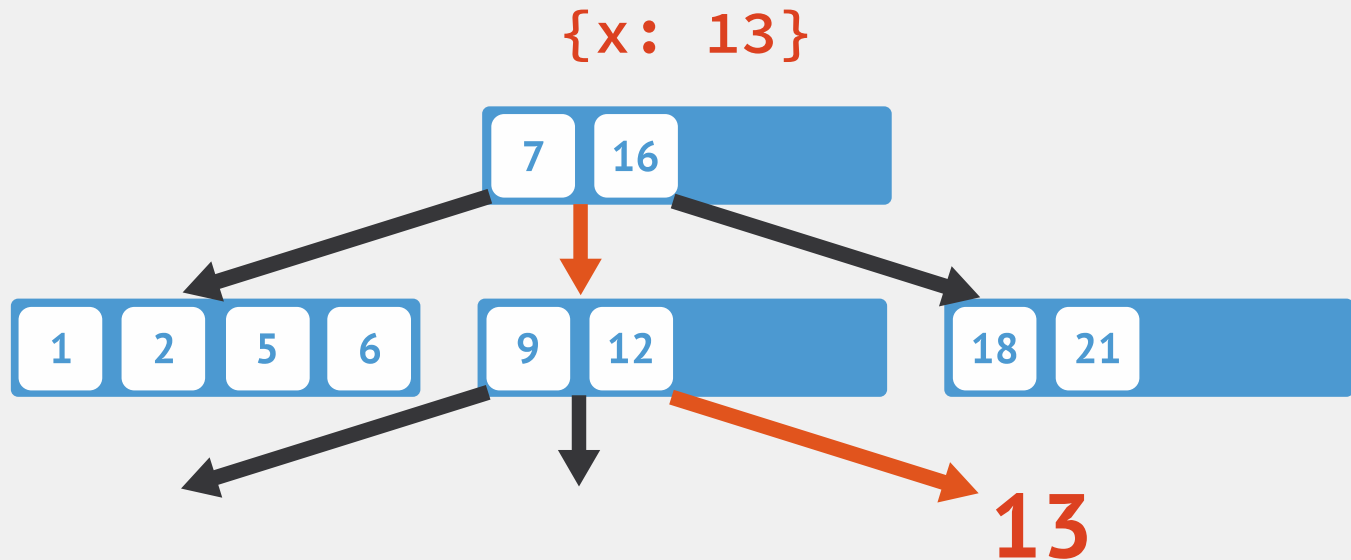
Finding 7 in Tree

B-Trees



Indexes in MongoDB are B-trees

B-Trees



Queries, inserts and deletes:
 $O(\log(n))$ time

Indexes are the single biggest
tunable performance factor in
MongoDB

Working with Indexes in MongoDB

How do I create indexes?

```
// Create an index if one does not exist  
// If an index exists throw an error  
db.recipes.createIndex({ main_ingredient: 1 })
```

```
// Create an index if it does not exists  
// Does not raise errors if index exists  
db.recipes.ensureIndex({ main_ingredient: 1 })
```

* 1 means ascending, -1 descending

What can be indexed?

// Multiple fields (compound indexes)

```
db.recipes.ensureIndex({  
  main_ingredient: 1,  
  calories: -1  
})
```

// Arrays of values (multikey indexes)

```
{  
  name: 'Chicken Noodle Soup',  
  ingredients : ['chicken', 'noodles']  
}
```

```
db.recipes.ensureIndex({ ingredients: 1 })
```

What can be indexed?

```
// Subdocuments
```

```
{  
  name : 'Apple Pie',  
  contributor: {  
    name: 'Joe American',  
    id: 'joea123'  
  }  
}
```

```
// Index field in a sub document
```

```
db.recipes.ensureIndex({ 'contributor.id': 1 })
```

How do I manage indexes?

// List a collection's indexes

```
db.recipes.getIndexes()
```

```
db.recipes.getIndexKeys()
```

// Drop a specific index

```
db.recipes.dropIndex({ ingredients: 1 })
```

// Drop all indexes and recreate them

```
db.recipes.reIndex()
```

// Default (unique) index on _id

Options

- Uniqueness constraints (unique, dropDups)
- Sparse Indexes
- Geospatial Indexes
 - 2d
 - 2dsphere (GeoJSON) 2.4+
- TTL Collections (expireAfterSeconds)

Limitations

- Collections can not have > 64 indexes.
- Index keys can not be > 1024 bytes (1K).
- The name of an index, including the namespace, must be < 128 characters.
- Queries can only use 1 index*
- Indexes have storage requirements, and impact the performance of writes.
- In memory sort (no-index) limited to 32mb of return data.

Optimize Your Queries

Profiling Slow Ops

```
db.setProfilingLevel( n , slowms=100ms )
```

n=0 profiler off

n=1 record operations longer than *slowms*

n=2 record all queries

```
db.system.profile.find()
```

* The profile collection is a capped collection

The Explain Plan (Pre Index)

```
db.recipes.find( { calories:
  { $lt : 40 } }
).explain( )
{
  "cursor" : "BasicCursor" ,
  "n" : 42,
  "nscannedObjects" : 12345
  "nscanned" : 12345,
  ...
  "millis" : 356,
  ...
}
```



Full Collection
Scan

* Doesn't use cached plans, re-evals and resets cache

The Explain Plan (Post Index)

```
db.recipes.find( { calories:  
  { $lt : 40 } }  
).{  
  "cursor" : "BtreeCursor calories_-1" ,  
  "n" : 42,  
  "nscannedObjects": 42  
  "nscanned" : 42,  
  ...  
  "millis" : 0,  
  ...  
}
```



Indexed Query

* Doesn't use cached plans, re-evals and resets cache

The Query Optimizer

- For each "type" of query, MongoDB periodically tries all useful indexes
- Aborts the rest as soon as one plan wins
- The winning plan is cached for each “type” of query
 - Up to 1000 writes
 - Change in indexes

Manually Select Index to Use

// Tell the database what index to use

```
db.recipes.find(  
  calories: { $lt: 1000 } }  
).hint({ _id: 1 })
```

// Tell the database to NOT use an index

```
db.recipes.find(  
  { calories: { $lt: 1000 } }  
).hint({ $natural: 1 })
```

Use Indexes to Sort Query Results

// Given the following index

```
db.collection.ensureIndex({ a:1, b:1 , c:1, d:1 })
```

// The following query and sort operations can use the index

```
db.collection.find( ).sort({ a:1 })
```

```
db.collection.find( ).sort({ a:1, b:1 })
```

```
db.collection.find({ a:4 }).sort({ a:1, b:1 })
```

```
db.collection.find({ b:5 }).sort({ a:1, b:1 })
```


Indexes that won't work for sorting query results

// Given the following index

```
db.collection.ensureIndex({ a:1, b:1, c:1, d:1 })
```

// These can not sort using the index

```
db.collection.find( ).sort({ b: 1 })
```

```
db.collection.find({ b: 5 }).sort({ b: 1 })
```

Covered Index Queries

// MongoDB can return data from just the index

```
db.recipes.ensureIndex({ main_ingredient: 1, name: 1 })
```

// Return only the ingredients field

```
db.recipes.find(  
  { main_ingredient: 'chicken' },  
  { _id: 0, name: 1 }  
)
```

// indexOnly will be true in the explain plan

```
db.recipes.find(  
  { main_ingredient: 'chicken' },  
  { _id: 0, name: 1 }  
)  
.explain()  
{  
  "indexOnly": true,  
}
```

Absent or suboptimal indexes
are the most common
avoidable MongoDB
performance problem.

Avoiding Common Mistakes

Trying to Use Multiple Indexes

// MongoDB can only use one index for a query

```
db.collection.ensureIndex({ a: 1 })
```

```
db.collection.ensureIndex({ b: 1 })
```

// Only one of the above indexes is used

```
db.collection.find({ a: 3, b: 4 })
```

Compound Key Mistakes

// Compound key indexes are very effective

```
db.collection.ensureIndex({ a: 1, b: 1, c: 1 })
```

// But only if the query is a prefix of the index

// This query can't use the index

```
db.collection.find({ c: 2 })
```

// ...but this query can

```
db.collection.find({ a: 3, b: 5 })
```

Low Selectivity Indexes

```
db.collection.distinct('status')  
[ 'new', 'processed' ]
```

```
db.collection.ensureIndex({ status: 1 })
```

// Low selectivity indexes provide little benefit

```
db.collection.find({ status: 'new' })
```

// Better

```
db.collection.ensureIndex({ status: 1, created_at: -1 })
```

```
db.collection.find(  
  { status: 'new' }  
)  
.sort({ created_at: -1 })
```

Regular Expressions

```
db.users.ensureIndex({ username: 1 })
```

```
// Left anchored regex queries can use the index
```

```
db.users.find({ username: /^joe smith/ })
```

```
// But not generic regexes
```

```
db.users.find({username: /smith/ })
```

```
// Or case insensitive queries
```

```
db.users.find({ username: /Joe/i })
```


Choosing the right indexes is one of the most important things you can do as a MongoDB developer so take the time to get your indexes right!



#codemash

Coming Next:

Part 3 : Replication and Sharding

Sridhar Nanjundeswaran

Engineer, MongoDB Inc.

@snanjund