# Indexes:

Single field

Compound

Multikey

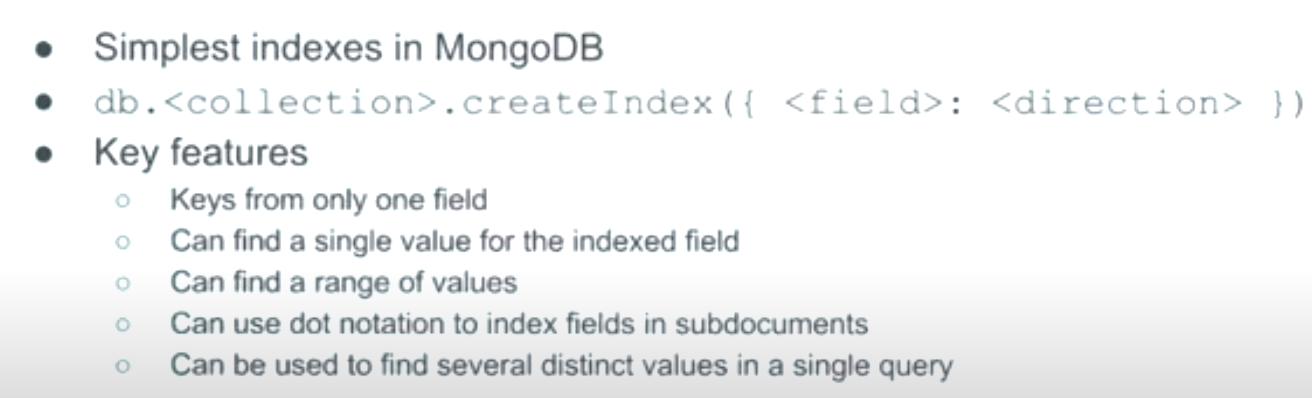
Multikey indexes collect and sort data from fields containing array values. Multikey indexes improve performance for queries on array fields.

You do not need to explicitly specify the multikey type. When you create an index on a field that contains an array value, MongoDB automatically sets that index to be a multikey index.

MongoDB can create multikey indexes over arrays that hold both scalar values (for example, strings and numbers) and embedded documents.

To create a multikey index, use the following prototype:

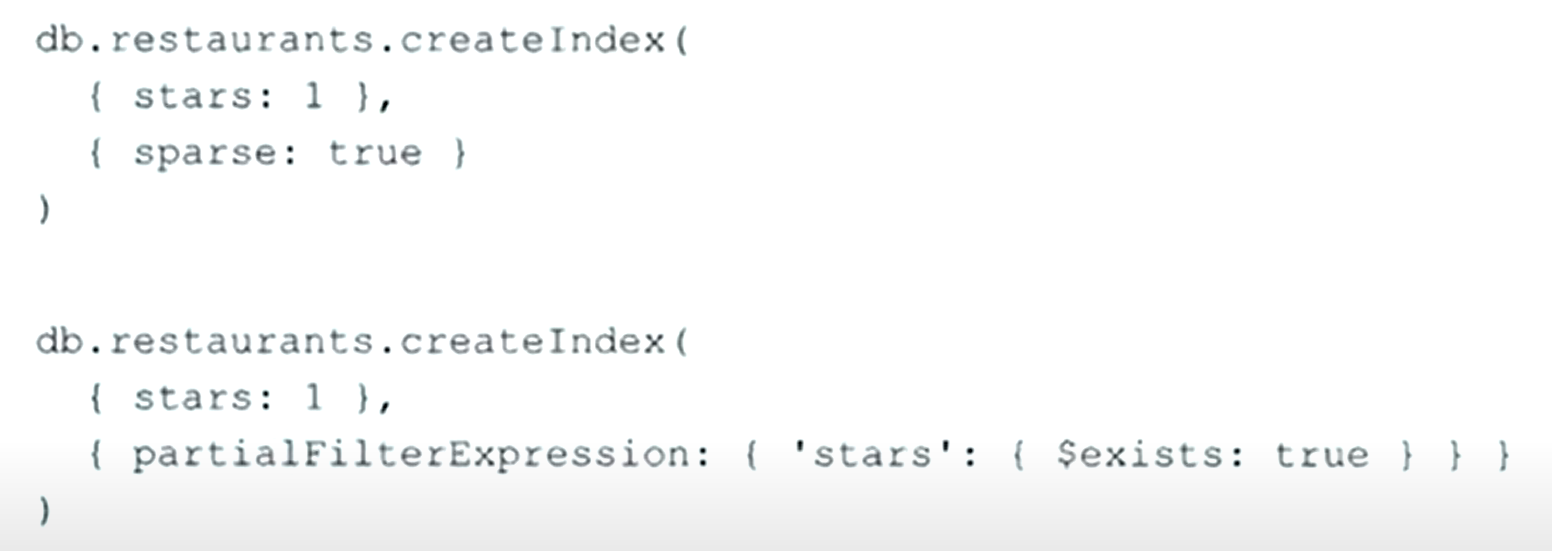
db.<collection>.createIndex( { <arrayField>: <sortOrder> } )



Index must contain at least 1 non array field. If you insert an entity with 2 indexed fields and both are arrays, it will throw an error.

Sparse index = index only where the field exists. NULL key not created

# Index resource allocations



# Key Words

$exists

$gt - greater

# Partial index

# Queries optimization

<https://www.youtube.com/watch?v=ApYCiVoLMlc&list=PL2iE4hzo22nmZOy23ACCmQhIVFmWQzF74&index=28>

# Covered queries – fully covered by index

Wildcards

MongoDB supports [flexible schemas](https://www.mongodb.com/docs/manual/core/data-modeling-introduction/#std-label-manual-data-modeling-intro), meaning document field names may differ within a collection. Use wildcard indexes to support queries against arbitrary or unknown fields.

To create a wildcard index, use the wildcard specifier ($\*\*) as the index key:

db.collection.createIndex( { **"$\*\*"**: <sortOrder> } )

Shard Key – поле, по которому шаржируем. Лучше, если есть во всех документах. Поле должно быть индексировано. Immutable, shard key can’t be changed in general and updated on specific document.

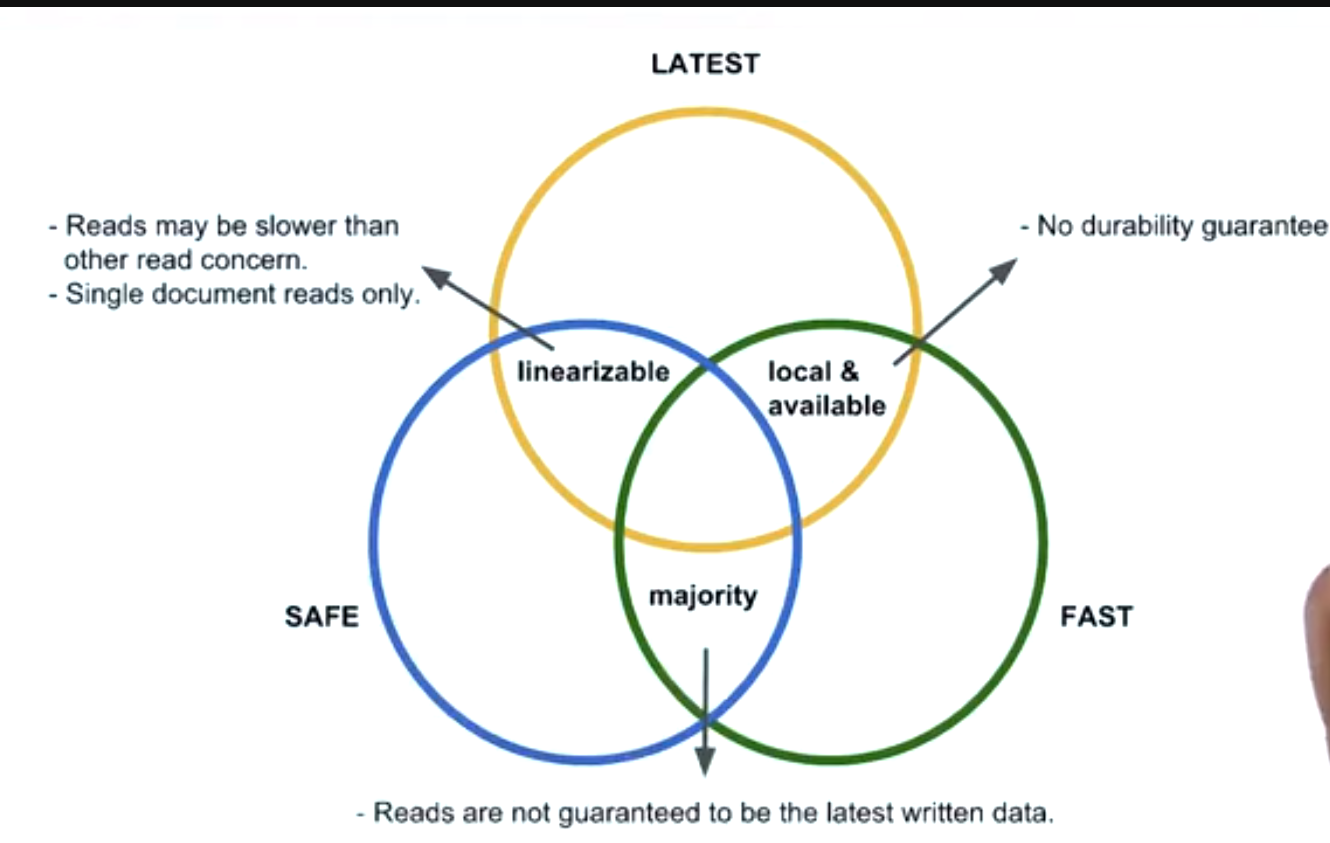
Replication

50 noeds max, 7 become primary, if more –

Secondary nodes:

Arbiters, hidden, delayed,

Read concerns



# Good shard key

Cardinality – the amount of possible unique values

Frequency – how often a unique value occurs

Monotonic change – avoid timestamps, dates, ids or use bounded (upper, lower boundary) keys limitations

Shard key is used for optimization on queries (like index)W

<https://www.youtube.com/watch?v=XX_XfTcEZY4&list=PL2iE4hzo22nmZOy23ACCmQhIVFmWQzF74&index=8>

Hashed shard keys

Can be used when there are some hotspots (frequent values). AS a result the data is more randomly distributed

The value can’t be an array

Looses the index performance boost

Collations

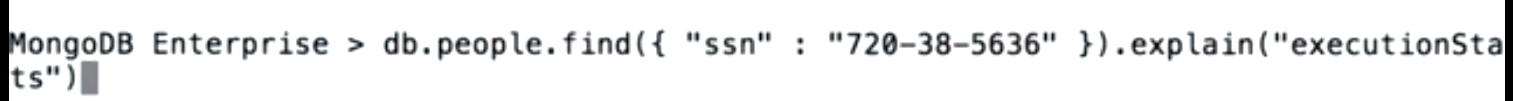
Language, case, sorting etc. Can be used to make case insensitive index

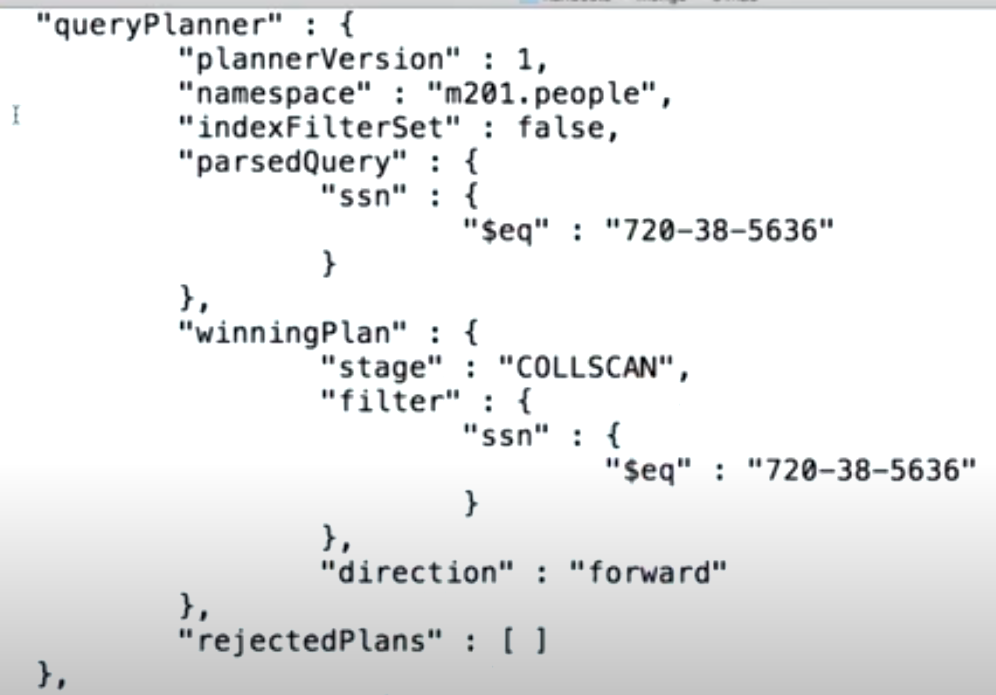
<https://www.youtube.com/watch?v=EAGNJvwBmWU>

<https://severalnines.com/blog/overview-mongodb-and-load-balancing/#:~:text=As%20MongoDB%20can%20handle%20concurrent,your%20read%20and%20write%20operations>.

IO parallelization

Execution plan:







Input stage – точка входа в каждую фазу. Читать в обратном порядке

# Performance in distributed systems

Cases

We needed to sort data regardless the case. I proposed to use index with collation that is case insensitive.  
<https://www.youtube.com/watch?v=aEXnHxk6CBU&list=PL2iE4hzo22nmZOy23ACCmQhIVFmWQzF74&index=13>