SD540 Server-Side Programming

Maharishi University of Management

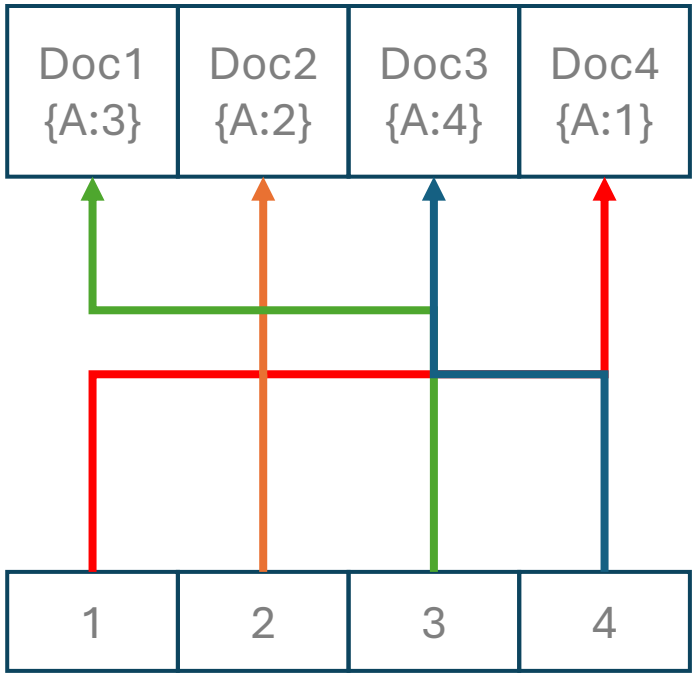
Masters of Software Development

Associate Professor Asaad Saad

Maharishi International University - Fairfield, Iowa

Index

- When we call **`find()`** a **full table/collection scan** will happen because our collection is not sorted which leads to slow performance.
- That's why we create an Index to boost the performance of our operations *(find, update, sort.. etc)*
- Indices live in memory and are constantly updated when a collection changes.

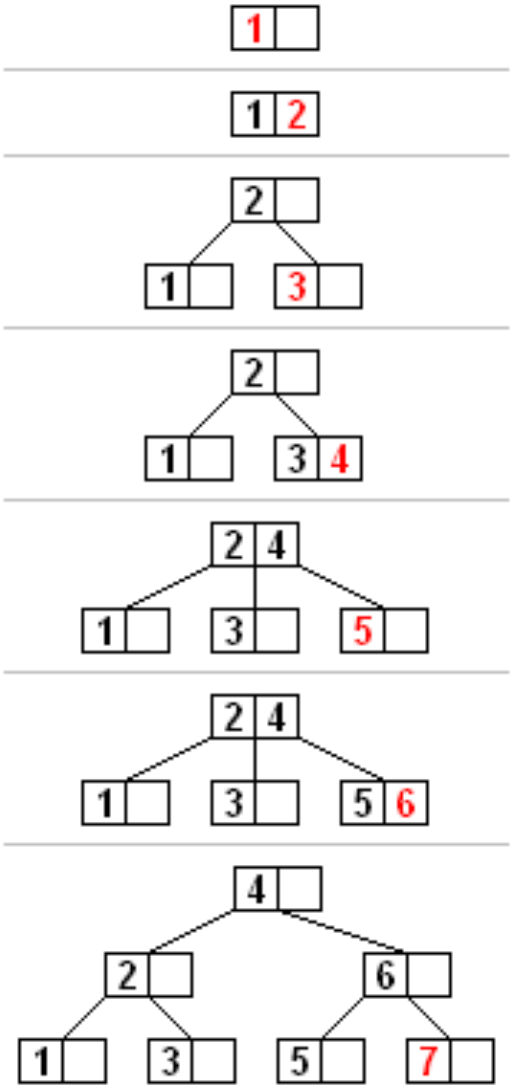


Index on A key

B-Tree Algorithm

MongoDB uses B-Tree algorithm to maintain indexes. B-tree is a self-balancing tree data structure that keeps data sorted and allows searches, sequential access, insertions, and deletions in logarithmic time.

Algorithm	**Average**	**Worst Case**
Search	O(log *n*)	O(log *n*)
Insert	O(log *n*)	O(log *n*)
Delete	O(log *n*)	O(log *n*)



Creating an Index

```
// { _id: 1, email:'asaad@miu.edu', password:'', ..}

// Create an index on email, sorted in ASC order

Model.createIndex({ email:1 })

findOne({ email: 'asaad@miu.edu' })
```

Notes:

- Creating an index will take time and space.
- Indexes should be created by a database administrator, as an optimization measure. It is not advised to create indexes from your application.
- The more indexes you have on a collection/DB, the more maintenance and memory it requires, and it may slow down the DB performance.

FullText Index

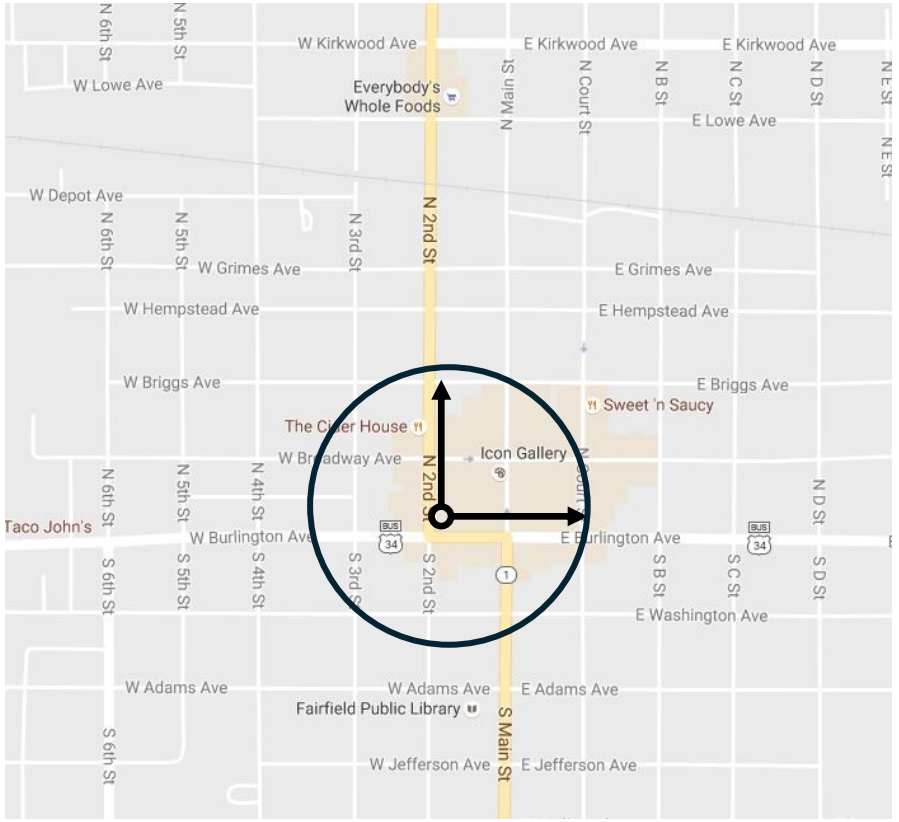
Solves the problem of searching for text. Regular expressions are very slow because they search the entire collection and compare the text against the provided expression which could be CPU intensive. Fulltext Index is the best way to build an index for all words.

```
// {_id: 1, title: 'Welcome to SD540 Class!'}

await Model.find({title: 'Welcome SD540 Class'}) // will find 0 docs
await Model.find({title: 'SD540'}) // will find 0 docs
await Model.find({title: {$regex: 'SD540'} }) // will find 1 doc but VERY slow!

Model.createIndex({title: 'text'})

// will find our document fast using the FullText index. spaces treated as OR
await Model.find({$text: {$search: 'Class SD540'} })
```

Geospatial Indexes (2D)

To find the nearest locations in MongoDB:

- Coordinates should be saved with `[long, lat]` format.
- Create a Geospatial Index (2d) on that key.
- Results are sorted by increasing distance, closest to furthest.




```
{_id: 1, name: 'Restaurant', location: [long,lat]})

Model.createIndex({location: '2d'})

await Model.find({location: {$near: [currentLong, currentLat]}}).limit(10)
```