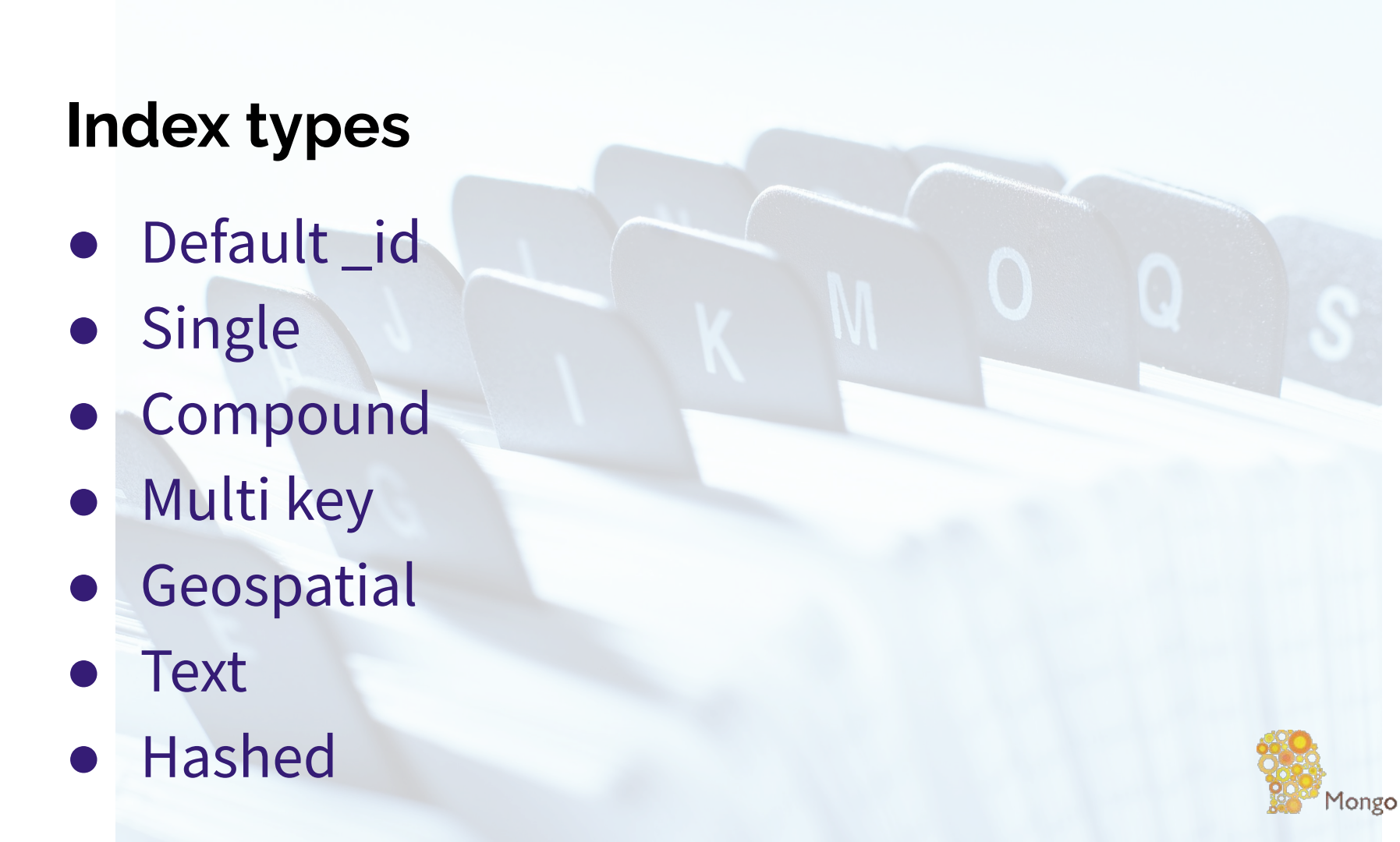
# Indexing

Ramesh S

# Index types

- Default _id
- Single
- Compound
- Multi key
- Geospatial
- Text
- Hashed



# Let's create a biggg collection

```
use big

for(i=0;i<1000000;i++){print(i);db.big.insert({a:i,b:i+2,c:i+4})}

db.big.count()
```

Mongo Factory

# Let's check its size

db.big.dataSize()

db.big.storageSize()

db.big.getIndexes()

Mongo Factory

# Let's create some indexes

```
db.big.createIndex({a:1})

db.big.createIndex({a:1,b:1},{background:true})

db.big.createIndex({a:1,b:1,c:1},{background:true})

db.big.createIndex({c:1},{unique:true})

db.big.getIndexes()
```

Mongo Factory

# Let's check the size of indexes

db.big.totalIndexSize()

db.big.dataSize()

db.big.totalSize()

db.big.storageSize()

Mongo Factory

# Checking how good an index is

db.big.find({a:89}).explain("executionStats")

Mongo Factory

# Let's Intersect them

db.big.find( {a: { $gt: 9 } , c: 14} )

Returns a single document having a=12, b=12 and c=14

Mongo Factory

# Working with indexes

db.big.getIndexes()

db.big.dropIndexes()

db.big.reIndex()

db.big.dropIndex(index)

Mongo Factory

# TTL indexes

```
db.events.createIndex( { "loggedAt": 1 }, { expireAfterSeconds: 864000} )

db.events.insert({“loggedAt”:new Date(),”mesg”:”Disk full”})

db.events.insert({“loggedAt”:new Date(),”mesg”:”Server not responding”})
```

Mongo Factory

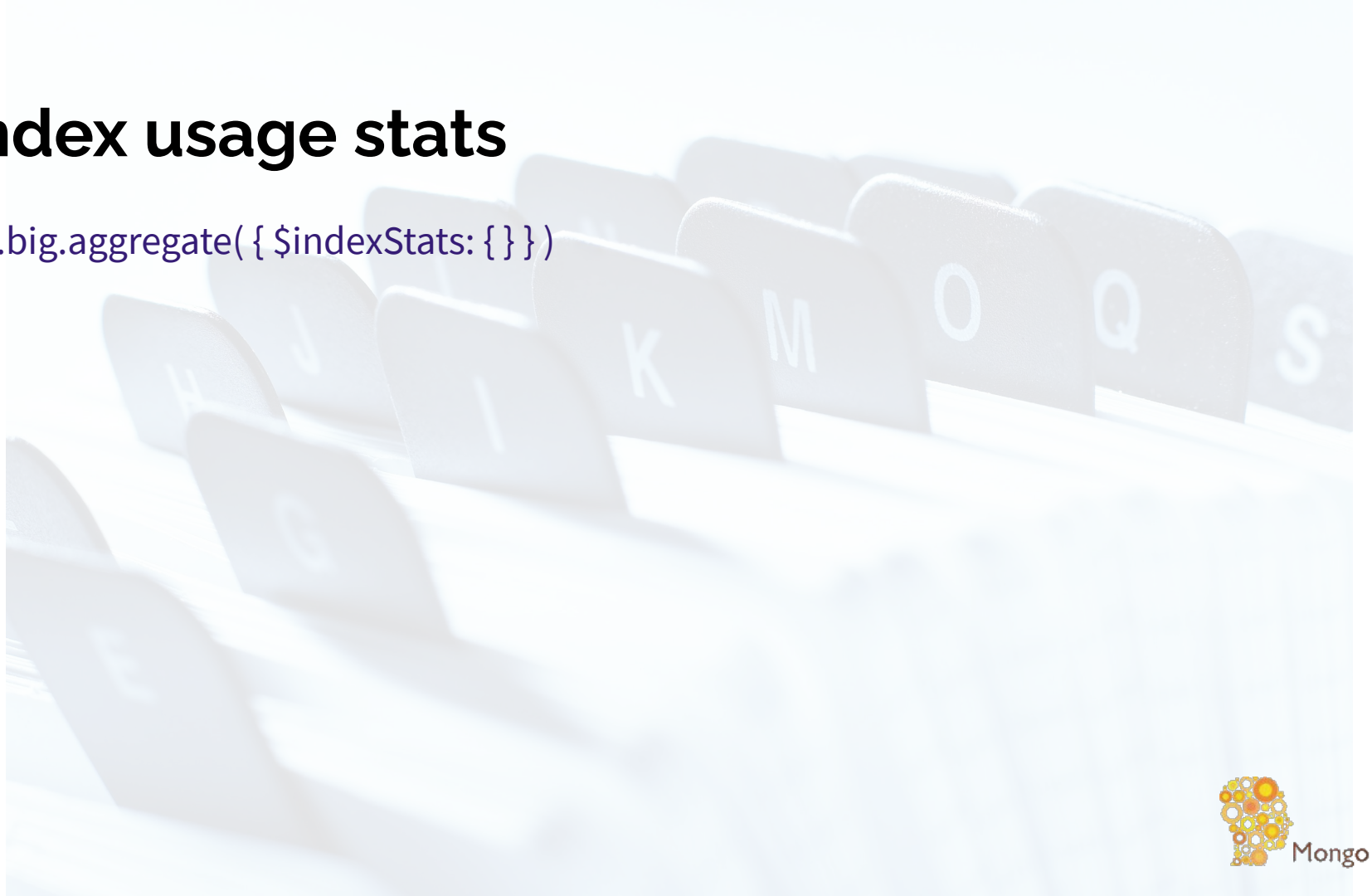
# TTL indexes

```
db.events.createIndex( { "eraseAt": 1 }, { expireAfterSeconds: 0} )

db.events.insert({“eraseAt”:new Date('June 30, 2016 14:00:00'),”mesg”:”Disk full”})

db.events.insert({“eraseAt”:new Date('June 30, 2016 14:00:00'),
                  ”mesg”:”Server not responding”})
```

Mongo Factory

# Index usage stats

```
db.big.aggregate( { $indexStats: { } } )
```



Mongo Factory

# GeoJSON Objects

To specify GeoJSON data, use an embedded document with:

- a field named type that specifies the GeoJSON object type and
- a field named coordinates that specifies the object's coordinates.
- If specifying latitude and longitude coordinates, list the longitude first and then latitude:
  - Valid longitude values are between -180 and 180, both inclusive.
  - Valid latitude values are between -90 and 90, both inclusive.

# GeoJSON Objects

```
location: {
        type: "Point",
        coordinates: [-73.856077, 40.848447]
}
```

Mongo Factory

# Legacy Coordinate Pairs

- To calculate distances on a Euclidean plane, store your location data as legacy coordinate pairs and use a 2d index.
- MongoDB supports spherical surface calculations on legacy coordinate pairs via a 2dsphere index by converting the data to the GeoJSON Point type.

# Legacy Coordinate Pairs

```
location: [-73.856077, 40.848447]
```

- Valid longitude values are between -180 and 180, both inclusive.
- Valid latitude values are between -90 and 90, both inclusive.

Mongo Factory

# Geospatial Indexes

MongoDB provides the following geospatial index types to support the geospatial queries.

- 2dsphere

  db.big.createIndex( { <location field> : "2dsphere" } )

  where the <location field> is a field whose value is either a GeoJSON object or a legacy coordinates pair.

- 2d

  db.collection.createIndex( { <location field> : "2d" } )

  where the <location field> is a field whose value is a legacy coordinates pair.

# Hashed Indexes

- Hashed indexes maintain entries with hashes of the values of the indexed field.
- Hashed indexes support sharding using hashed shard keys.
- Hashed based sharding uses a hashed index of a field as the shard key to partition data across your sharded cluster.
- Using a hashed shard key to shard a collection results in a more random distribution of data.

# Hashed Indexes

To create a hashed index, specify hashed as the value of the index key

```
db.big.createIndex( { _id: "hashed" } )
```

Mongo Factory

# Hashed Indexes

- MongoDB supports hashed indexes of any single field.
- The hashing function collapses embedded documents and computes the hash for the entire value, but does not support multi-key (i.e. arrays) indexes.
- You may not create compound indexes that have hashed index fields or specify a unique constraint on a hashed index;
- You can create both a hashed index and an ascending/descending (i.e. non-hashed) index on the same field: MongoDB will use the scalar index for range queries.

# List all Hashed Indexes

```
db.adminCommand("listDatabases").databases.forEach(function(d){
   let mdb = db.getSiblingDB(d.name);
   mdb.getCollectionInfos({ type: "collection" }).forEach(function(c){
      let currentCollection = mdb.getCollection(c.name);
      currentCollection.getIndexes().forEach(function(idx){
        let idxValues = Object.values(Object.assign({}, idx.key));

        if (idxValues.includes("hashed")) {
          print("Hashed index: " + idx.name + " on " + idx.ns);
          printjson(idx);
        };
      });
   });
});
```

Mongo Factory