MongoDB 3.0

Indexes

Topics

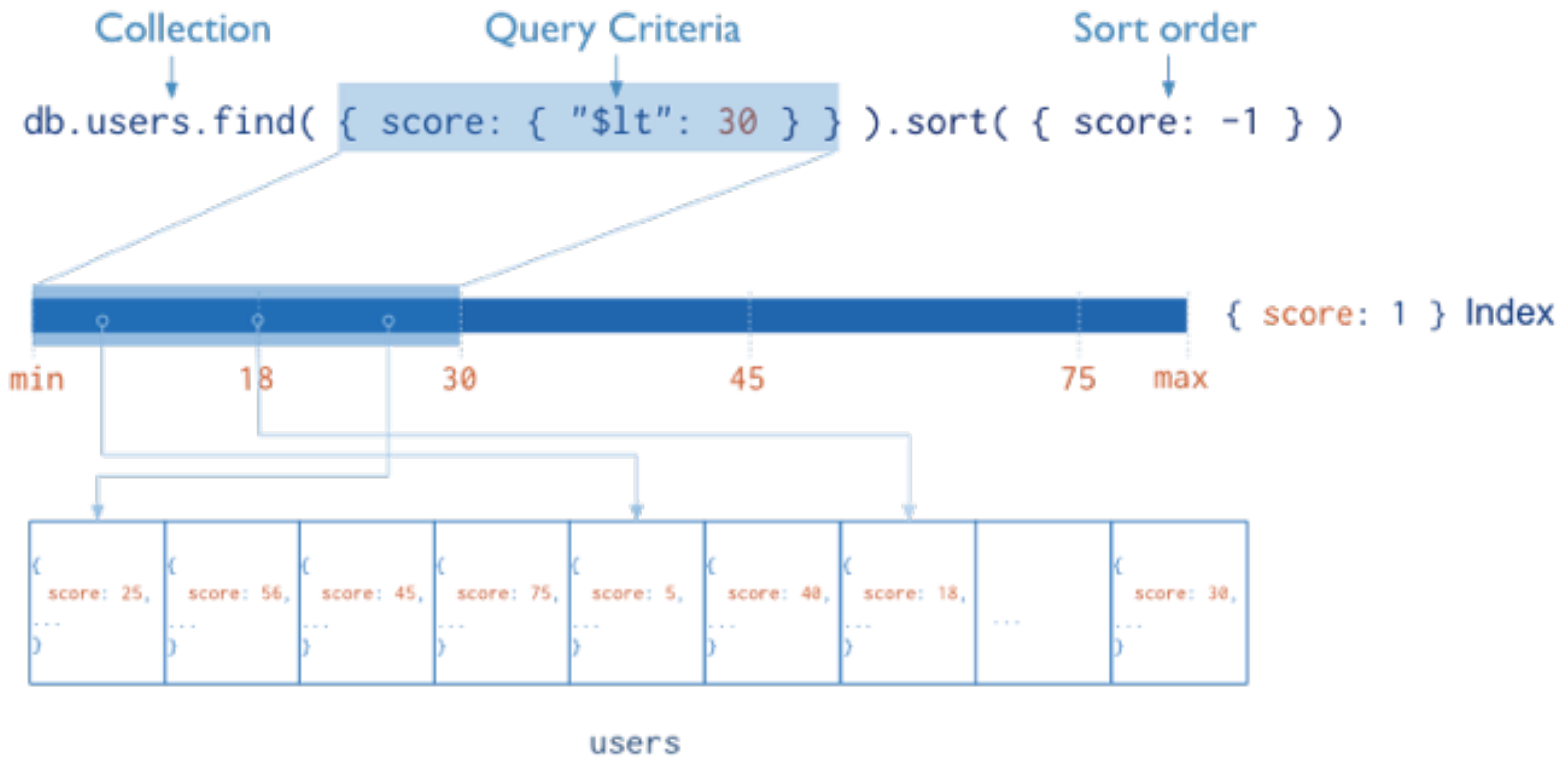
- MongoDB Indexes
- Index Types
- Index Properties
- Manage Indexes

Indexes

- Indexes support the efficient execution of queries in MongoDB.
- Without indexes, MongoDB must perform a collection scan, i.e. scan every document in a collection, to select those documents that match the query statement.
- If an appropriate index exists for a query, MongoDB can use the index to limit the number of documents it must inspect.
- Indexes are special data structures that store a small portion of the collection's data set in an easy to traverse form. The index stores the value of a specific field or set of fields, ordered by the value of the field.
- The ordering of the index entries supports efficient equality matches and range-based query operations. In addition, MongoDB can return sorted results by using the ordering in the index.

Indexes

- The following diagram illustrates a query that selects and orders the matching documents using an index:

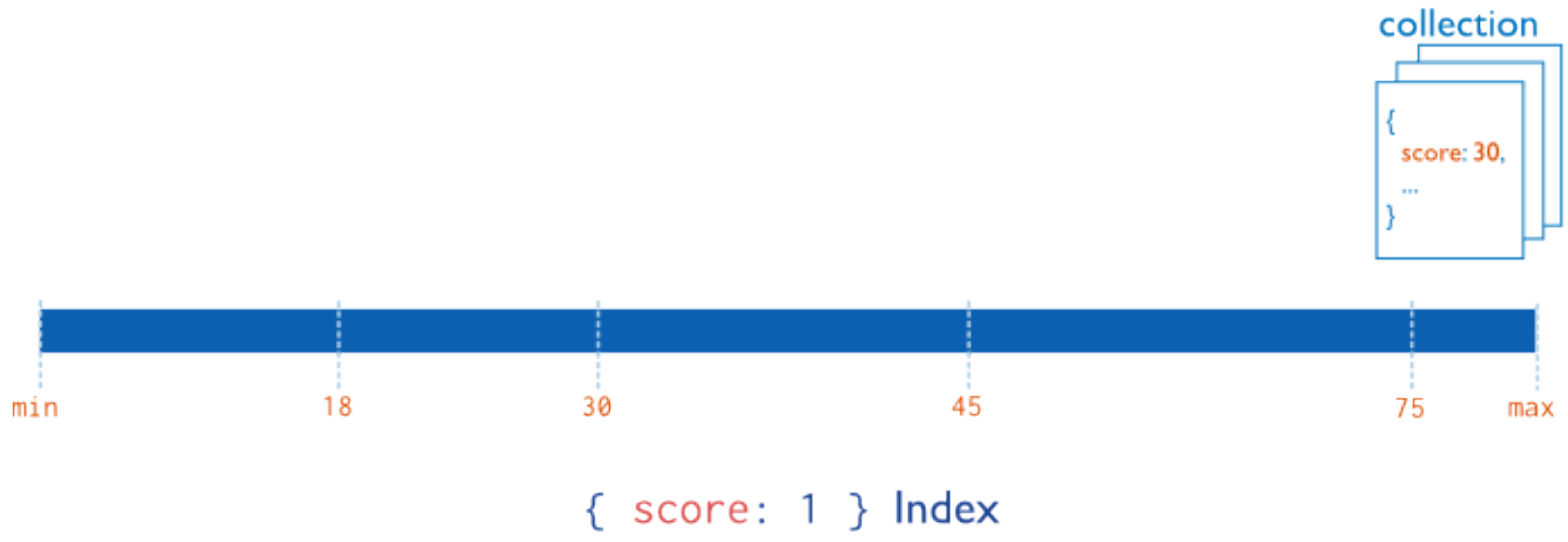


- Fundamentally, indexes in MongoDB are similar to indexes in other database systems. MongoDB defines indexes at the collection level and supports indexes on any field or sub-field of the documents in a MongoDB collection.

Index Types

★ Default _id

- All MongoDB collections have an index on the `_id` field that exists by default. If applications do not specify a value for _id the driver or the mongod will create an _id field with an ObjectId value.
- The _id index is unique and prevents clients from inserting two documents with the same value for the _id field.
- MongoDB supports the creation of user-defined ascending/descending indexes on a single field of a document.

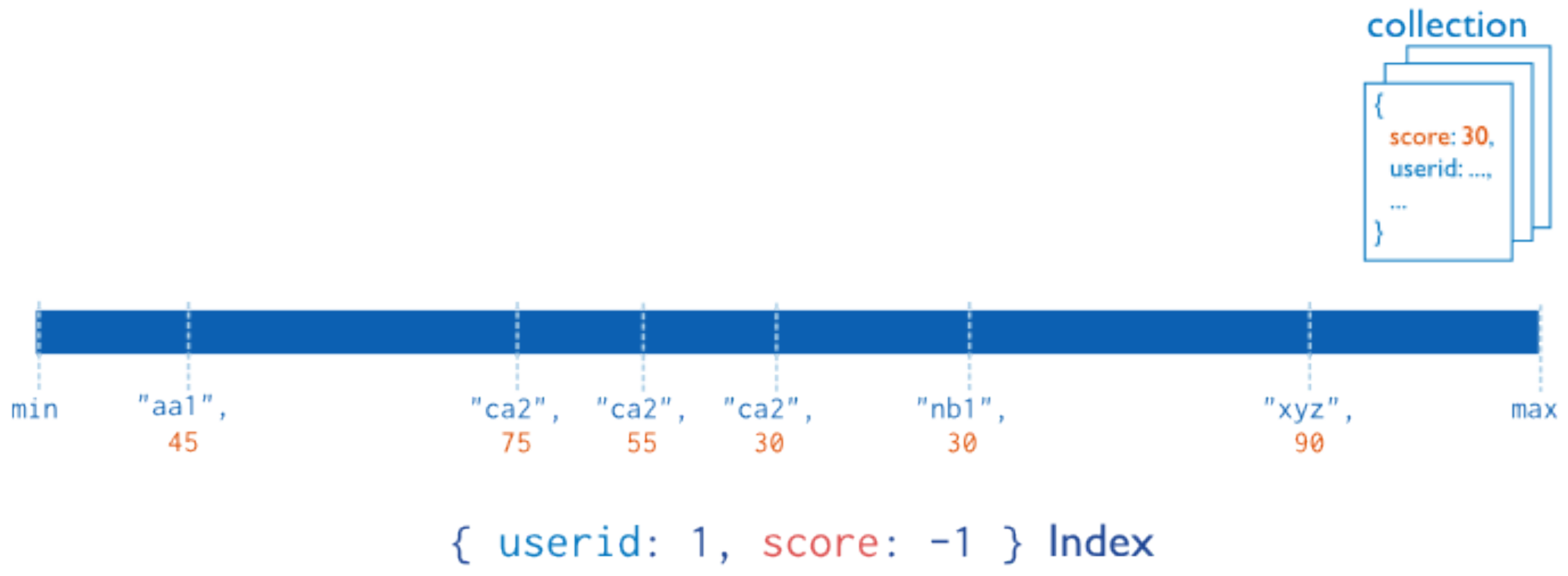


Index Types

★ Compound Index

- MongoDB also supports user-defined indexes on multiple fields, i.e. compound indexes.

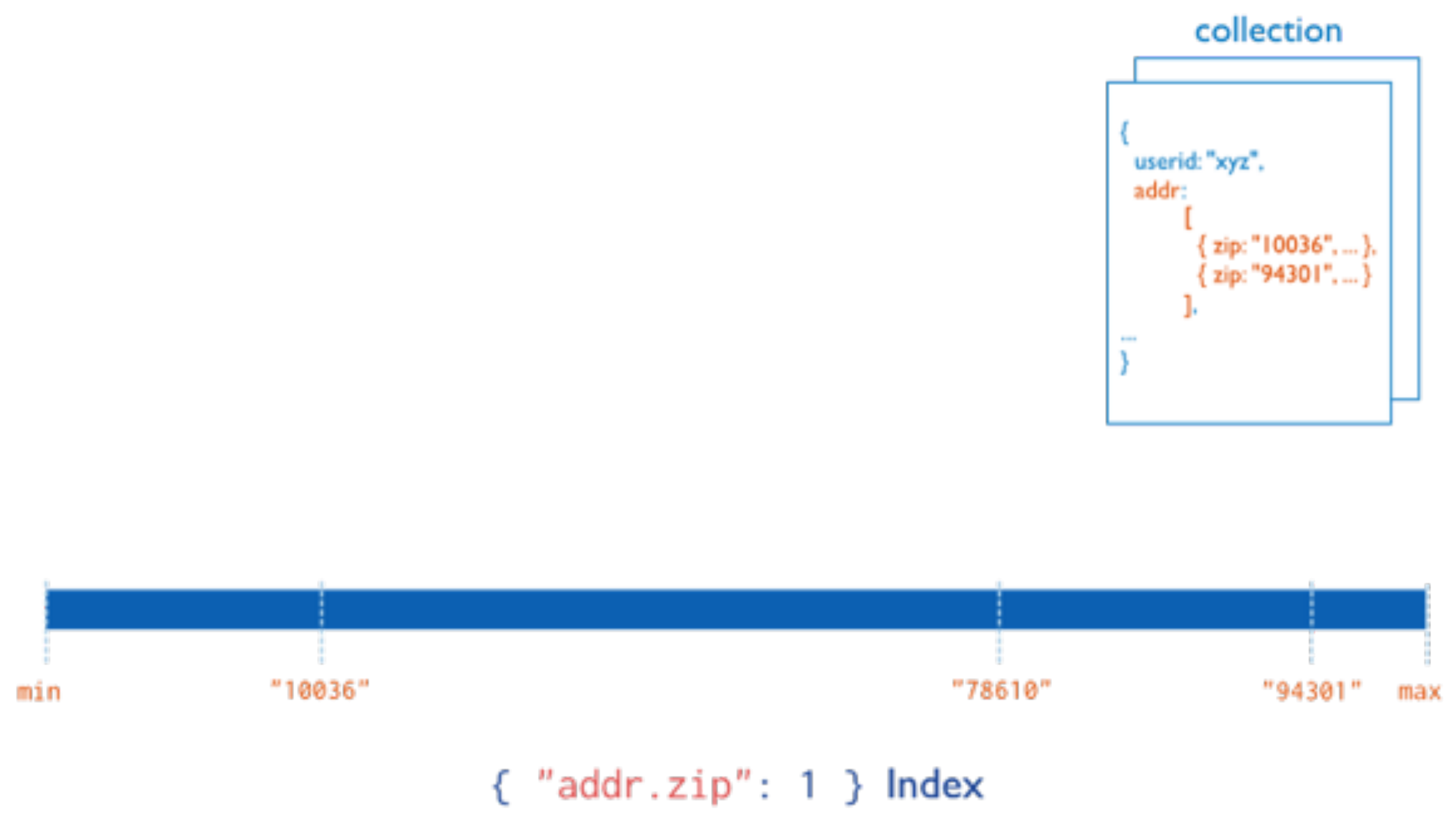
- The order of fields listed in a compound index has significance. For instance, if a compound index consists of `{ userid: 1, score: -1 }`, the index sorts first by userid and then, within each userid value, sorts by score.



Index Types

★ Multikey Index

- MongoDB uses multi-key indexes to index the content stored in arrays.
- If you index a field that holds an array value, MongoDB creates separate index entries for every element of the array. These multi-key indexes allow queries to select documents that contain arrays by matching on element or elements of the arrays.
- MongoDB automatically determines whether to create a multi-key index if the indexed field contains an array value; you do not need to explicitly specify the multi-key type.



Index Types

★ Others

- Geospatial Index: To support efficient queries of geospatial coordinate data, MongoDB provides two special indexes: 2d indexes that uses planar geometry when returning results and 2sphere indexes that use spherical geometry to return results.

- Text Indexes: MongoDB provides a text index type that supports searching for string content in a collection. These text indexes do not store language-specific stop words (e.g. “the”, “a”, “or”) and stem the words in a collection to only store root words.

- Hashed Indexes: To support hash based sharding, MongoDB provides a hashed index type, which indexes the hash of the value of a field. These indexes have a more random distribution of values along their range, but only support equality matches and cannot support range-based queries.

Index Properties

- Unique Indexes:

  The unique property for an index causes MongoDB to reject duplicate values for the indexed field. To create a unique index on a field that already has duplicate values, see index-creation-duplicate-dropping for index creation options. Other than the unique constraint, unique indexes are functionally interchangeable with other MongoDB indexes.

- Sparse Indexes:

  The sparse property of an index ensures that the index only contain entries for documents that have the indexed field. The index skips documents that do not have the indexed field.

  You can combine the sparse index option with the unique index option to reject documents that have duplicate values for a field but ignore documents that do not have the indexed key.

- TTL Indexes:

  TTL indexes are special indexes that MongoDB can use to automatically remove documents from a collection after a certain amount of time. This is ideal for certain types of information like machine generated event data, logs, and session information that only need to persist in a database for a finite amount of time.

Create Index

- Create an Index on a Single Field

  ```
  db.records.createIndex( { userid: 1 } )
  ```

- The value of the field in the index specification describes the kind of index for that field. For example, a value of 1 specifies an index that orders items in ascending order. A value of -1 specifies an index that orders items in descending order.

Create Index

- Create a Compound Index

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

- The value of the field in the index specification describes the kind of index for that field. For example, a value of 1 specifies an index that orders items in ascending order. A value of -1 specifies an index that orders items in descending order.

Create Index

- Create a Unique Index on a Single Field

```
db.collection.createIndex({a:1},{unique:true})
```

- Unique Compound Index

```
db.collection.createIndex({a:1,b:1},{unique:true})
```

- Unique Index and Missing Field

```
db.collection.createIndex({a:1},{unique:true,sparse:true})
```

Remove Index

- To remove an index from a collection use the dropIndex() method and the following procedure. If you simply need to rebuild indexes you can use the process described in the Rebuild Indexes document.

- Remove a Specific Index

```
db.accounts.dropIndex( { "tax-id": 1 } )
```

- Remove All Indexes (except for the `_id`)

```
db.collection.dropIndexes()
```

- To modify an existing index, you need to drop and recreate the index.

Rebuild Index

- If you need to rebuild indexes for a collection you can use the db.collection.reIndex() method to rebuild all indexes on a collection in a single operation.

- This operation drops all indexes, including the _id index, and then rebuilds all indexes.

  ```
  db.accounts.reIndex()
  ```

References

- Index Introduction
  http://docs.mongodb.org/manual/core/indexes-introduction/

- Indexes Tutorials
  http://docs.mongodb.org/manual/tutorial/create-an-index/

- More references
  http://docs.mongodb.org/manual/reference/indexes/