Mongo DB Introduction

Agenda

- What is MongoDB?
- Key Features
- What is a Document?
- DataBases and Collections

MongoDB

- **MongoDB is an open-source document database that provides**
  - high performance,
  - high availability, and
  - automatic scaling.

Key Features

- **MongoDB provides high performance data persistence. In particular,**
  - Support for embedded data models reduces I/O activity on database system.
  - Indexes support faster queries and can include keys from embedded documents and arrays.

Rich Query Language

- **MongoDB supports a rich query language to support read and write operations (CRUD) as well as:**
  - Data Aggregation
  - Text Search and Geospatial Queries.

High Availability

- **MongoDB’s replication facility, called replica set, provides:**
  - automatic failover and
  - data redundancy.
- **A replica set is a group of MongoDB servers that maintain the same data set, providing redundancy and increasing data availability.**

Horizontal Scalability

- **MongoDB provides horizontal scalability as part of its core functionality:**
  - Sharding distributes data across a cluster of machines.
  - MongoDB 3.4 supports creating zones of data based on the shard key.
  - In a balanced cluster, MongoDB directs reads and writes covered by a zone only to those shards inside the zone.

What is a Document?

- A record in MongoDB is a document, made up of field and value pairs.
- MongoDB documents are similar to JSON objects.
- The values of fields may include other documents, arrays, and arrays of documents.

What is a Document?

```
{
  name: "sue",                          ← field: value
  age: 26,                              ← field: value
  status: "A",                          ← field: value
  groups: [ "news", "sports" ]          ← field: value
}
```

Advantages of Document

- Documents (i.e. objects) correspond to native data types in many programming languages.
- Embedded documents and arrays reduce need for expensive joins.
- Dynamic schema supports fluent polymorphism.

Databases and Collections

- **MongoDB stores**
  - BSON documents, i.e. data records, in collections;
  - The maximum BSON document size is 16 megabytes.
  - the collections in databases.

Create a Database

- **Databases hold collections of documents.**
- **If a database does not exist, MongoDB creates the database when you first store data for that database.**
  - use myDB

Collections

- **MongoDB stores documents in collections.**
- **Like tables in relational databases.**
  - db.myNewCollection2.insertOne({ x: 1 })
- **The insertOne() operation creates both the database myNewDB and the collection myNewCollection2 if they do not already exist.**

User Commands

- Query and Write Operation Commands
- Aggregation Commands
- User Management Commands
- Role Management Commands

Query and Write Operation Commands

Description
delete
Deletes one or more documents.
eval
Deprecated. Runs a JavaScript function on the database server.
find
Selects documents in a collection or a view.
findAndModify
Returns and modifies a single document.
getLastError
Returns the success status of the last operation.
getMore
Returns batches of documents currently pointed to by the cursor.
getPrevError
Returns status document containing all errors since the last resetError command.
insert
Inserts one or more documents.
parallelCollectionScan
Lets applications use multiple parallel cursors when reading documents from a collection.
resetError
Resets the last error status.
update
Updates one or more documents.

1) insert

- **The insert command inserts one or more documents and returns a document containing the status of all inserts.**

- insert

- {
- insert: <collection>,
- documents: [<document>, <document>, <document>, ...],
- ordered: <boolean>,
- writeConcern: { <write concern> },
- bypassDocumentValidation: <boolean>
- }

Insert Options

- **Ordered**
  - If true, then when an insert of a document fails, return without inserting any remaining documents listed in the inserts array.
- **WriteConcern**
  - Write concern describes the level of acknowledgement requested from MongoDB for write operations to a standalone mongod or to replica sets or to sharded clusters.
- **BypassDocumentValidation**
  - This lets you insert documents that do not meet the validation requirements.

Insert

- db.runCommand(
- {
- insert: "users",
- documents: [{ _id: 1, user: "abc123", status: "A" }]
- }
-)

insert

- Bulk Insert
- db.runCommand(
- {
- insert: "users",
- documents: [
- { _id: 2, user: "ijk123", status: "A" },
- { _id: 3, user: "xyz123", status: "P" },
- { _id: 4, user: "mop123", status: "P" }
-],
- ordered: false,
- writeConcern: { w: "majority", wtimeout: 5000 }
- }
-)

2) update

- The update command modifies documents in a collection.
- A single update command can contain multiple update statements.

update

```
• {
•   update: <collection>,
•   updates: [
•     { q: <query>, u: <update>, upsert: <boolean>, multi: <boolean>,
•       collation: <document>, arrayFilters: <array> },
•     { q: <query>, u: <update>, upsert: <boolean>, multi: <boolean>,
•       collation: <document>, arrayFilters: <array> },
•     { q: <query>, u: <update>, upsert: <boolean>, multi: <boolean>,
•       collation: <document>, arrayFilters: <array> },
•     ...
•   ],
•   ordered: <boolean>,
•   writeConcern: { <write concern> },
•   bypassDocumentValidation: <boolean>
• }
```

Update options

- **q<query>**
  - The query that matches documents to update. Use the same query selectors as used in the find() method.
- **u: <update>**
  - The modifications to apply.
- **Upsert**
  - If true, perform an insert if no documents match the query.
  - If both upsert and multi are true and no documents match the query, the update operation inserts only a single document.

Update options

- **Multi**
  - If true, updates all documents that meet the query criteria.
  - If false, limit the update to one document that meet the query criteria.
  - Defaults to false.
- **Collation**
  - Collation allows users to specify language-specific rules for string comparison, such as rules for lettercase and accent marks.

Update Specific Fields of One Document

```
db.runCommand(
  {
    update: "users",
    updates: [
      {
        q: { user: "abc123" }, u: { $set: { status: "A" }, $inc: { points: 1 } }
      }
    ],
    ordered: false,
    writeConcern: { w: "majority", wtimeout: 5000 }
  }
)
```

Update Operators

- Name Description
- $currentDate Sets the value of a field to current date, either as a Date or a Timestamp.
- $inc Increments the value of the field by the specified amount.
- $min Only updates the field if the specified value is less than the existing field value.
- $max Only updates the field if the specified value is greater than the existing field value.
- $mul Multiplies the value of the field by the specified amount.
- $rename Renames a field.
- $set Sets the value of a field in a document.
- $setOnInsert Sets the value of a field if an update results in an insert of a document. Has no effect on update operations that modify existing documents.
- $unset Removes the specified field from a document.

Update Specific Fields of Multiple Documents

```
db.runCommand(
  {
    update: "users",
    updates: [
      { q: { }, u: { $set: { status: "A" }, $inc: { points: 1 } }, multi: true }
    ],
    ordered: false,
    writeConcern: { w: "majority", wtimeout: 5000 }
  }
)
```