In this chapter, we will learn the different operations with Arangosh.

The following are the possible operations with Arangosh −

* Creating a Document Collection
* Creating Documents
* Reading Documents
* Updating Documents

Let us start by creating a new database. We will use the following line of code to create a new database −

127.0.0.1:8529@\_system> db.\_createDatabase("song\_collection")

true

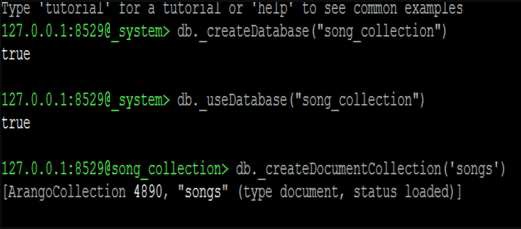
The following line of code will help you shift to the new database −

127.0.0.1:8529@\_system> db.\_useDatabase("song\_collection")

true

Prompt will shift to "@@song\_collection"

127.0.0.1:8529@song\_collection>



From here we will study CRUD Operations. Let us create a collection into the new database −

127.0.0.1:8529@song\_collection> db.\_createDocumentCollection('songs')

Output

[ArangoCollection 4890, "songs" (type document, status loaded)]

127.0.0.1:8529@song\_collection>

Let us add a few documents (JSON objects) to our 'songs' collection.

We add the first document in the following way −

127.0.0.1:8529@song\_collection> db.songs.save({title: "A Man's Best Friend",

lyricist: "Johnny Mercer", composer: "Johnny Mercer", Year: 1950, \_key:

"A\_Man"})

Output

{

"\_id" : "songs/A\_Man",

"\_key" : "A\_Man",

"\_rev" : "\_VjVClbW---"

}

Let us add other documents to the database. This will help us learn the process of querying the data. You can copy these codes and paste the same in Arangosh to emulate the process −

127.0.0.1:8529@song\_collection> db.songs.save(

{

title: "Accentchuate The Politics",

lyricist: "Johnny Mercer",

composer: "Harold Arlen", Year: 1944,

\_key: "Accentchuate\_The"

}

)

{

"\_id" : "songs/Accentchuate\_The",

"\_key" : "Accentchuate\_The",

"\_rev" : "\_VjVDnzO---"

}

127.0.0.1:8529@song\_collection> db.songs.save(

{

title: "Affable Balding Me",

lyricist: "Johnny Mercer",

composer: "Robert Emmett Dolan",

Year: 1950,

\_key: "Affable\_Balding"

}

)

{

"\_id" : "songs/Affable\_Balding",

"\_key" : "Affable\_Balding",

"\_rev" : "\_VjVEFMm---"

}

How to Read Documents

The **\_key** or the document handle can be used to retrieve a document. Use document handle if there is no need to traverse the collection itself. If you have a collection, the document function is easy to use −

127.0.0.1:8529@song\_collection> db.songs.document("A\_Man");

{

"\_key" : "A\_Man",

"\_id" : "songs/A\_Man",

"\_rev" : "\_VjVClbW---",

"title" : "A Man's Best Friend",

"lyricist" : "Johnny Mercer",

"composer" : "Johnny Mercer",

"Year" : 1950

}

How to Update Documents

Two options are available to update the saved data − **replace** and **update**.

The update function patches a document, merging it with the given attributes. On the other hand, the replace function will replace the previous document with a new one. The replacement will still occur even if completely different attributes are provided. We will first observe a non-destructive update, updating the attribute Production` in a song −

127.0.0.1:8529@song\_collection> db.songs.update("songs/A\_Man",{production:

"Top Banana"});

Output

{

"\_id" : "songs/A\_Man",

"\_key" : "A\_Man",

"\_rev" : "\_VjVOcqe---",

"\_oldRev" : "\_VjVClbW---"

}

Let us now read the updated song's attributes −

127.0.0.1:8529@song\_collection> db.songs.document('A\_Man');

Output

{

"\_key" : "A\_Man",

"\_id" : "songs/A\_Man",

"\_rev" : "\_VjVOcqe---",

"title" : "A Man's Best Friend",

"lyricist" : "Johnny Mercer",

"composer" : "Johnny Mercer",

"Year" : 1950,

"production" : "Top Banana"

}

A large document can be easily updated with the **update** function, especially when the attributes are very few.

In contrast, the **replace** function will abolish your data on using it with the same document.

127.0.0.1:8529@song\_collection> db.songs.replace("songs/A\_Man",{production:

"Top Banana"});

Let us now check the song we have just updated with the following line of code −

127.0.0.1:8529@song\_collection> db.songs.document('A\_Man');

Output

{

"\_key" : "A\_Man",

"\_id" : "songs/A\_Man",

"\_rev" : "\_VjVRhOq---",

"production" : "Top Banana"

}

Now, you can observe that the document no longer has the original data.

How to Remove Documents

The remove function is used in combination with the document handle to remove a document from a collection −

127.0.0.1:8529@song\_collection> db.songs.remove('A\_Man');

Let us now check the song's attributes we just removed by using the following line of code −

127.0.0.1:8529@song\_collection> db.songs.document('A\_Man');

We will get an exception error like the following as an output −

JavaScript exception in file

'/usr/share/arangodb3/js/client/modules/@arangodb/arangosh.js' at 97,7:

ArangoError 1202: document not found

! throw error;

! ^

stacktrace: ArangoError: document not found

at Object.exports.checkRequestResult

(/usr/share/arangodb3/js/client/modules/@arangodb/arangosh.js:95:21)

at ArangoCollection.document

(/usr/share/arangodb3/js/client/modules/@arangodb/arango-collection.js:667:12)

at <shell command>:1:10

