

INTRODUCTION TO CASSANDRA

UNIT -2

Chapter 7

Sowmya V
Assistant Professor
BMSCE, Bangalore

Collections - Set

To update the table “student_info” to provide the values for “hobbies” for the student with Rollno =1.

```
UPDATE student_info  
    SET hobbies = hobbies + {'Chess, Table Tennis'}  
    WHERE RollNo=1;
```

To confirm the values in the hobbies column, use the below command:

```
SELECT *  
    FROM student_info  
    WHERE RollNo=1;
```

```
cqlsh:students> select * from student_info where RollNo=1;
```

rollno	dateofjoining	hobbies	language	lastexampercent	studname
1	2012-03-29 00:00:00India Standard Time	{'Chess, Table Tennis'}	null	69.6	Michael Storm

(1 rows)

CQL Data Definition Commands

- **CREATE KEYSPACE** – Creates a KeySpace in Cassandra.
- **USE** – Connects to a created KeySpace.
- **ALTER KEYSPACE** – Changes the properties of a KeySpace.
- **DROP KEYSPACE** – Removes a KeySpace
- **CREATE TABLE** – Creates a table in a KeySpace.
- **ALTER TABLE** – Modifies the column properties of a table.
- **DROP TABLE** – Removes a table.
- **TRUNCATE** – Removes all the data from a table.
- **CREATE INDEX** – Defines a new index on a single column of a table.
- **DROP INDEX** – Deletes a named index.

CQL Data Manipulation Commands

- **INSERT** – Adds columns for a row in a table.
- **UPDATE** – Updates a column of a row.
- **DELETE** – Deletes data from a table.
- **BATCH** – Executes multiple DML statements at once.

CQL Clauses

- **SELECT** – This clause reads data from a table
- **WHERE** – The where clause is used along with select to read a specific data.
- **ORDERBY** – The orderby clause is used along with select to read a specific data in a specific order.



Data Types



1

**Built-In
Data Type**

2

**Collection
Data Type**

3

**User-Defined
Data Type**



Data Type	Constants	Description
ascii	strings	Represents ASCII character string
bigint	bigint	Represents 64-bit signed long
blob	blobs	Represents arbitrary bytes
Boolean	booleans	Represents true or false
counter	integers	Represents counter column
decimal	integers, floats	Represents variable-precision decimal
double	integers	Represents 64-bit IEEE-754 floating point
float	integers, floats	Represents 32-bit IEEE-754 floating point
inet	strings	Represents an IP address, IPv4 or IPv6
int	integers	Represents 32-bit signed int
text	strings	Represents UTF8 encoded string
timestamp	integers, strings	Represents a timestamp
timeuuid	uuids	Represents type 1 UUID
uuid	uuids	Represents type 1 or type 4
		UUID
varchar	strings	Represents UTF8 encoded string
varint	integers	Represents arbitrary-precision integer



2

Collection Data Type

Collection	Description
list	A list is a collection of one or more ordered elements.
map	A map is a collection of key-value pairs.
set	A set is a collection of one or more elements.

3

User-Defined Data Type

- **CREATE TYPE** – Creates a user-defined datatype.
- **ALTER TYPE** – Modifies a user-defined datatype.
- **DROP TYPE** – Drops a user-defined datatype.
- **DESCRIBE TYPE** – Describes a user-defined datatype.
- **DESCRIBE TYPES** – Describes user-defined datatypes.

Cassandra - COLLECTIONS



Cassandra - COLLECTIONS

When to use collection?

Use collection when it is required to store or denormalize a small amount of data.

What is the limit on the values of items in a collection?

The values of items in a collection are limited to 64K.

Where to use collections?

Collections can be used when you need to store the following:

1. Phone numbers of users.
2. Email ids of users.

Collections - Set

Set is a data type that is used to store a group of elements.

Creating a Table with Set

Objective: To create a table “users” with an “emails” column. The type of this column “emails” is “set”.

Act:

```
CREATE TABLE  users(  
user_id text primary key,  
first_name text,  
last_name text,  
emails set<text>  
);
```

Collections - Set

Inserting Data into a Set

Objective: To insert values into the “emails” column of the “users” table.

Note: Set value must be unique

Act:

```
INSERT INTO  users
(user_id, first_name, last_name, emails)
VALUES('AB','Albert','Baggins',{'a@baggings.com',
'baggins@gmail.com'});
```

Collections - Set

Updating a Set

Objective: Add an element to a set using the UPDATE command and the addition(+) operator.

Act:

UPDATE users

**SET emails = emails + {'ab@friendsofmordor.org'}
WHERE user_id = 'AB';**

Collections - Set

Verification / READ

Objective: To retrieve email addresses for Albert from the set.

Act:

```
SELECT user_id, emails
FROM users
WHERE user_id = 'AB';
```

Outcome:

```
cqlsh:students> SELECT user_id, emails FROM users WHERE user_id = 'AB';
```

user_id	emails
AB	{'a@baggins.com', 'ab@friendsofmordor.org', 'baggins@gmail.com'}

Collections - Set

DELETE

Objective: To remove an element from a set using the subtraction(-) operator.

Act:

UPDATE users

**SET emails = emails - {'ab@friendsofmordor.org'}
WHERE user_id = 'AB';**

Collections - Set

DELETE

Objective: To remove all elements from a set using the UPDATE or DELETE at statement.

Act:

UPDATE users

SET emails = {}

WHERE user_id = 'AB';

```
sqlsh:students> select * from users;
```

user_id	emails	first_name	last_name
AB	null	Albert	Baggins

```
(1 rows)
```

Collections - Set

DELETE

Objective: To remove all elements from a set using the UPDATE or DELETE at statement.

Act:

**DELETE emails
FROM users
WHERE user_id = 'AB';**

```
cqlsh:students> select * from users;  
user_id | emails | first_name | last_name  
-----+-----+-----+-----  
      AB | null  |    Albert |   Baggins  
(1 rows)
```

Collections - Set

Objective: To alter the schema for the table “student_info” to add a column “hobbies”.

Act:

```
ALTER TABLE student_info ADD hobbies set<text>;
```

Collections - List

List is used in the cases where

- the order of the elements is to be maintained, and
- a value is to be stored multiple times.

Collections - List

Creating a Table with List

Objective: To create a table “users” with an “emails” column. The type of this column “list” is “list”.

Act:

```
CREATE TABLE  users1(  
user_id text primary key,  
first_name text,  
last_name text,  
emails list<text>  
);
```

Collections - List

Objective: To alter the schema for the table “student_info” to add a list column “language”.

Act:

```
ALTER TABLE student_info ADD language list<text>;
```

Confirm the structure of the table after the change has been made:

```
cqlsh:students> describe table student_info;
```

```
CREATE TABLE student_info (  
  rollno int,  
  dateofjoining timestamp,  
  hobbies set<text>,  
  language list<text>,  
  lastexampercent double,  
  studname text,  
  PRIMARY KEY (rollno)  
) WITH  
  bloom_filter_fp_chance=0.010000 AND  
  caching='KEYS_ONLY' AND  
  comment='' AND  
  dclocal_read_repair_chance=0.000000 AND  
  gc_grace_seconds=864000 AND  
  index_interval=128 AND  
  read_repair_chance=0.100000 AND  
  replicate_on_write='true' AND  
  populate_io_cache_on_flush='false' AND  
  default_time_to_live=0 AND  
  speculative_retry='NONE' AND  
  memtable_flush_period_in_ms=0 AND  
  compaction={'class': 'SizeTieredCompactionStrategy'} AND  
  compression={'sstable_compression': 'LZ4Compressor'};  
  
CREATE INDEX student_info_lastexampercent_idx ON student_info (lastexampercent);  
  
CREATE INDEX student_info_studname_idx ON student_info (studname);  
  
cqlsh:students>
```

Collections - List

Objective: To update values in the list column, “language” of the table “student_info”.

UPDATE student_info

SET language = language + ['Hindi, English']

WHERE RollNo=1;

```
cqlsh:students> select rollNo, studname, hobbies, language from student_info;
```

rollno	studname	hobbies	language
1	Michael Storm	{ 'Chess, Table Tennis' }	{ 'Hindi, English' }
4	Ian String	{ 'Lawn Tennis, Table Tennis, Golf' }	{ 'Hindi, English' }
3	David Flemming	{ 'Chess, Badminton' }	{ 'Hindi, English, French' }

(3 rows)

Collections - List

Objective: To alter the table “users” and to add a column “top_places” of type list.

Act:

```
ALTER TABLE users ADD “top_places” list<text>;
```

```
cqlsh:students> describe table users;
```

```
CREATE TABLE users (  
  user_id text,  
  emails set<text>,  
  first_name text,  
  last_name text,  
  top_places list<text>,  
  PRIMARY KEY (user_id)  
) WITH  
  bloom_filter_fp_chance=0.010000 AND  
  caching='KEYS_ONLY' AND  
  comment='' AND  
  dclocal_read_repair_chance=0.000000 AND  
  gc_grace_seconds=864000 AND  
  index_interval=128 AND  
  read_repair_chance=0.100000 AND  
  replicate_on_write='true' AND  
  populate_io_cache_on_flush='false' AND  
  default_time_to_live=0 AND  
  speculative_retry='NONE' AND  
  memtable_flush_period_in_ms=0 AND  
  compaction={'class': 'SizeTieredCompactionStrategy'} AND  
  compression={'sstable_compression': 'LZ4Compressor'};
```

Collections - List

Objective: To update the list column “top_places” in the “users” table for user_id = ‘AB’.

Act:

```
UPDATE users
    SET top_places = [ 'Lonavla', 'Khandala' ]
    WHERE user_id = 'AB';
```

Outcome:

```
cqlsh:students> select * from users where user_id = 'AB';
```

user_id	emails	first_name	last_name	top_places
AB	null	Albert	Baggins	['Lonavla', 'Khandala']

```
(1 rows)
```

Collections - List

Objective: Prepend an element to the list by enclosing it in square brackets and using the addition (+) operator.

Act:

```
UPDATE users
```

```
SET top_places = [ 'Mahabaleshwar' ] + top_places
```

```
WHERE user_id = 'AB';
```

Outcome:

```
cqlsh:students> select * from users;
```

user_id	emails	first_name	last_name	top_places
AB	null	Albert	Baggins	['Mahabaleshwar', 'Lonavla', 'Khandaia']

Collections - List

Objective: To remove an element from a list using the DELETE command and the list index position in square brackets.

The record as it exists prior to deletion is

```
cqlsh:students> SELECT user_id, top_places FROM users WHERE user_id = 'AB';
```

user_id	top_places
AB	['Mahabaleshwar', 'Lonavla', 'Khandala', 'Tapola']

(1 rows)

Act:

```
DELETE top_places[3]  
FROM users  
WHERE user_id = 'AB';
```

Outcome: The status after deletion is

```
cqlsh:students> select * from users;
```

user_id	emails	first_name	last_name	top_places
AB	null	Albert	Baggins	['Mahabaleshwar', 'Lonavla', 'Khandala']

Collections – Map: Key, Value Pair

Map is a data type that is used to store a key-value pair of elements.

To alter the “users” table to add a map column “todo”.

```
ALTER TABLE users
```

```
ADD todo map<timestamp, text>;
```

Outcome:

```
cqlsh:students> describe table users;
```

```
CREATE TABLE users (  
  user_id text,  
  emails set<text>,  
  first_name text,  
  last_name text,  
  todo map<timestamp, text>,  
  top_places list<text>,  
  PRIMARY KEY (user_id)  
) WITH  
  bloom_filter_fp_chance=0.010000 AND  
  caching='KEYS_ONLY' AND  
  comment='' AND  
  dclocal_read_repair_chance=0.000000 AND  
  gc_grace_seconds=864000 AND  
  index_interval=128 AND  
  read_repair_chance=0.100000 AND  
  replicate_on_write='true' AND  
  populate_io_cache_on_flush='false' AND  
  default_time_to_live=0 AND  
  speculative_retry='NONE' AND  
  memtable_flush_period_in_ms=0 AND  
  compaction={'class': 'SizeTieredCompactionStrategy'} AND  
  compression={'sstable_compression': 'LZ4Compressor'};
```

Collections – Map: Key,Value Pair

To update the record for user (user_id = 'AB') in the “users” table.

UPDATE users

SET todo =

**{ '2014-9-24': 'Cassandra Session', '2014-10-2 12:00' :
'MongoDB Session' }**

WHERE user_id = 'AB';

Outcome:

```
|cqlsh:students> select user_id, todo from users where user_id='AB';
```

user_id	todo
AB	{'2014-09-24 00:00:00India Standard Time': 'Cassandra Session', '2014-10-02 12:00:00India Standard Time': 'MongoDB Session'}

Collections – Map: Key,Value Pair

To delete the record in the “users” table.

```
DELETE todo['2014-9-24']  
FROM users  
WHERE user_id = 'AB';
```

Outcome:

```
cqlsh:students> select user_id, todo from users where user_id='AB';
```

user_id	todo
AB	{'2014-10-02 12:00:00India Standard Time': 'MongoDB Session'}

```
(1 rows)
```

USING A COUNTER

A Counter is a special column that is changed in increments.

Eg: Books issued in the library

USING A COUNTER

Step1: Create a table library_book with the column names counter_value(counter), book_name(varchar), stud_name(varchar) and PRIMARY KEY = book_name and stud_name

Step 2: Load data into the counter column along with book_name “ Fundamentals of Business Analytics” and stud_name as “Jeet”

Step 3: Take a look at the counter value

Step 4: Increase the counter value with the same book_name and stud_name = “Shaan”

Step 5: Again, Take a look at the counter value

Step 6: Update another record for stud_name “Jeet”

Step 7: Verify the counter value after update.

Step 1:

```
CREATE TABLE library_book (  
    counter_value counter,  
    book_name varchar,  
    stud_name varchar,  
    PRIMARY KEY (book_name, stud_name)  
);
```

Step 2: Load data into the counter column.

```
UPDATE library_book  
SET counter_value = counter_value + 1  
WHERE book_name='Fundamentals of Business Analytics' AND stud_name='jeet';
```

Step 3: Take a look at the counter value.

SELECT *

FROM library_book;

Output is:

```
cqlsh:students> select * from library_book;
```

book_name	stud_name	counter_value
Fundamentals of Business Analytics	jeet	1

(1 rows)

Step 4: Let us increase the value of the counter.

UPDATE library_book

SET counter_value = counter_value + 1

WHERE book_name='Fundamentals of Business Analytics' AND stud_name='shaan';

```
cqlsh:students> UPDATE library_book
... SET counter_value = counter_value + 1
... WHERE book_name='Fundamentals of Business Analytics' AND stud_name='shaan';
```

Step 5: Again, take a look at the counter value.

```
cqlsh:students> select * from library_book;
```

book_name	stud_name	counter_value
Fundamentals of Business Analytics	jeet	1
Fundamentals of Business Analytics	shaan	1

(2 rows)

Step 6: Update another record for Stud_name "Jeet".

UPDATE library_book

SET counter_value = counter_value + 1

WHERE book_name='Fundamentals of Business Analytics' AND stud_name='jeet';

```
cqlsh:students> UPDATE library_book
... SET counter_value = counter_value + 1
... WHERE book_name='Fundamentals of Business Analytics' AND stud_name='jeet';
```

Step 7: Let us take a look at the counter value, one last time.

```
cqlsh:students> select * from library_book;
```

book_name	stud_name	counter_value
Fundamentals of Business Analytics	jeet	2
Fundamentals of Business Analytics	shaan	1

(2 rows)

Time To Live

Data in a column, other than a counter column, can have an optional expiration period called TTL (time to live). The client request may specify a TTL value for the data. The TTL is specified in seconds.

```
CREATE TABLE userlogin(  
  userid int primary key, password text  
);
```

```
INSERT INTO userlogin (userid, password) VALUES (1,'infy') USING TTL 30;
```

```
SELECT TTL (password)  
  FROM userlogin  
 WHERE userid=1;
```

```
cqlsh:students> SELECT TTL (password)  
                  ... FROM userlogin  
                  ... WHERE userid=1;  
  
  ttl(password)  
-----  
                  18
```


ALTER COMMANDS

1. Create a table “sample” with columns “sample_id” and “sample_name”
2. Insert a record into table “sample”
3. View the record of the table “sample”

ALTER COMMANDS

```
CREATE TABLE sample(  
    sample_id text,  
    sample_name text,  
    primary key(sample_id)  
);
```

```
INSERT INTO sample (sample_id, sample_name) VALUES ('S101',BIG DATA');
```

```
SELECT *  
    FROM sample;
```

```
cqlsh:students> select * from sample;  
  
sample_id | sample_name  
-----+-----  
      S101 |      Big Data  
  
(1 rows)
```

ALTER TABLE to Change the Data Type of a column

1. Alter the schema of the table “sample”. Change the data type of the column “sample_id” to integer from text.

```
ALTER TABLE sample  
ALTER sample_id TYPE int;
```

2. Insert value ‘S102’ and ‘Big Data’ as sample_id and ‘sample_name’

```
INSERT INTO sample(sample_id, sample_name)  
VALUES( 'S102', 'Big Data');
```

```
INSERT INTO sample(sample_id, sample_name)  
VALUES( 102, 'Big Data' );
```

```
cqlsh:students> Insert into sample(sample_id, sample_name) values( 102, 'Big Data' );  
cqlsh:students> select * from sample;
```

sample_id	sample_name
1395732529	Big Data
102	Big Data

4. Alter the data type of the “sample_id” column to varchar from integer.

```
ALTER TABLE sample
```

```
ALTER sample_id TYPE varchar;
```

```
cqlsh:students> alter table sample alter sample_id type varchar;
```

5. Check the records after the data type of “sample_id” has been changed to varchar from integer.

```
cqlsh:students> select * from sample;
```

sample_id	sample_name
S101	Big Data
\x00\x00\x00f	Big Data

(2 rows)

ALTER TABLE to DELETE a column

1. Drop the column “sample_id” from the table “sample”.

```
ALTER TABLE sample  
DROP sample_id;
```

```
|cqlsh:students> alter table sample drop sample_id;  
Bad Request: Cannot drop PRIMARY KEY part sample_id
```

2. Drop the column “sample_name” from the table “sample”.

```
ALTER TABLE sample  
DROP sample_name;
```

```
|cqlsh:students> alter table sample drop sample_name;
```

7.10.3 Drop a Table

1. Drop the column family/table “sample”.

DROP columnfamily sample;

```
|cqlsh:students> drop columnfamily sample;
```

The above request succeeds. The table/column family no longer exists in the keyspace.

2. Confirm the non-existence of the table “sample” in the keyspace by giving the following command:

```
cqlsh:students> describe table sample;
```

```
column family 'sample' not found
```

7.10.4 Drop a Database

1. Drop the keyspace “students”.

DROP keyspace students;

```
|cqlsh:students> drop keyspace students;
```

2. Confirm the non-existence of the keyspace “students” by issuing the following command:

```
cqlsh:students> describe keyspace students;
```

Keyspace 'students' not found.

IMPORT AND EXPORT

Export data to a CSV file

Export the contents of the table/column family “elearninglists” present in the “students” database to a CSV file (d:\elearninglists.csv).

Step 1: Check the records of the table “elearninglists” present in the “students” database.

```
SELECT *  
FROM elearninglists;
```

```
cqlsh:students> select * from elearninglists;
```

id	course_order	course_id	courseowner	title
101	1	1001	Subhashini	NoSQL Cassandra
101	2	1002	Seema	NoSQL MongoDB
101	3	1003	Seema	Hadoop Sqoop
101	4	1004	Subhashini	Hadoop Flume

(4 rows)

Export data to a CSV file

Step 2: Execute the below command at the cqlsh prompt:

COPY elearninglists (id, course_order, course_id, courseowner, title) TO 'd:\elearninglists.csv';

Import data from a CSV file

To import data from “D:\elearninglists.csv” into the table “elearninglists” present in the “students” database.

COPY elearninglists (id, course_order, course_id, courseowner, title) FROM 'd:\elearninglists.csv';

```
SELECT *  
FROM elearninglists;
```

```
cqlsh:students> select * from elearninglists;
```

id	course_order	course_id	courseowner	title
101	1	1001	Subhashini	NoSQL Cassandra
101	2	1002	Seema	NoSQL MongoDB
101	3	1003	Seema	Hadoop Sqoop
101	4	1004	Subhashini	Hadoop Flume

```
(4 rows)
```

```
cqlsh:students>
```

Import from STDIN

Step 1: Ensure that the table “persons” exists in the database “students”.

DESCRIBE TABLE persons;

```
cqlsh:students> describe table persons;
```

```
CREATE TABLE persons (  
  id int,  
  fname text,  
  lname text,  
  PRIMARY KEY (id)  
) WITH  
  bloom_filter_fp_chance=0.010000 AND  
  caching='KEYS_ONLY' AND  
  comment='' AND  
  dclocal_read_repair_chance=0.000000 AND  
  gc_grace_seconds=864000 AND  
  index_interval=128 AND  
  read_repair_chance=0.100000 AND  
  replicate_on_write='true' AND  
  populate_io_cache_on_flush='false' AND  
  default_time_to_live=0 AND  
  speculative_retry='NONE' AND  
  memtable_flush_period_in_ms=0 AND  
  compaction={'class': 'SizeTieredCompactionStrategy'} AND  
  compression={'sstable_compression': 'LZ4Compressor'};
```

Step 2:

```
COPY persons (id, fname, lname) FROM STDIN;
```

```
cqlsh:students> COPY persons (id, fname, lname) FROM STDIN;  
[Use \. on a line by itself to end input]  
[copy] 1,"Samuel","Jones"  
[copy] 2,"Virat","Kumar"  
[copy] 3,"Andrew","Simon"  
[copy] 4,"Raul","A Simpson"  
[copy] \.
```

```
SELECT *  
FROM persons;
```

```
cqlsh:students> select * from persons;
```

id	fname	lname
1	Samuel	Jones
2	Virat	Kumar
4	Raul	A Simpson
3	Andrew	Simon

Export to STDOUT

Step 1: Check the records of the table “elearninglists” present in the “students” database.

```
SELECT *  
FROM elearninglists;
```

```
cqlsh:students> select * from elearninglists;
```

id	course_order	course_id	courseowner	title
101	1	1001	Subhashini	NoSQL Cassandra
101	2	1002	Seema	NoSQL MongoDB
101	3	1003	Seema	Hadoop Sqoop
101	4	1004	Subhashini	Hadoop Flume

Export to STDOUT

Step 2: Execute the below command at the cqlsh prompt.

COPY elearninglists (id, course_order, course_id, courseowner, title) TO STDOUT;

```
cqlsh:students> copy elearninglists (id, course_order, course_id, courseowner, title) to STDOUT;  
101,1,1001,Subhashini,NoSQL Cassandra  
101,2,1002,Seema,NoSQL MongoDB  
101,3,1003,Seema,Hadoop Sqoop  
101,4,1004,Subhashini,Hadoop Flume  
4 rows exported in 0.031 seconds.  
cqlsh:students>
```

ASSIGNMENT 1: COLLECTIONS

Objective: To learn about the various collection types: Set, List and Map.

Problem Description: Design a table/column family to support the following requirements.

- Store the basic information about students such as Student Roll No, Student Name, Student Date of Birth, and Student Address.
- Store the subject preferences of each student. There should be a minimum of two subject preferences and a maximum of four. The order of preferences as given by the student should be preserved.
- Store the hobbies of each student. There should be a minimum of two hobbies and a maximum of four. The hobbies as given by the student should be arranged in alphabetical order.

ASSIGNMENT 2: TIME TO LIVE

Objective: To learn about the TTL type (Time To Live).

Problem Description: Design a table/column family to support the following requirements.

Store the login details of the user such as UserID and Password. The information stored should expire in a day's time.

ASSIGNMENT 3: IMPORT FROM CSV

Objective: To learn about the import from CSV to Cassandra table/column family.

Problem Description: Read a public dataset from the site www.kdnuggets.com. If not already in CSV format, first convert to CSV format and then import into a Cassandra table/column family by the name "PublicDataSet" in the "Sample" database.

Confirm the presence of data in the table "PublicDataSet" in the "Sample" database.

Further Readings

- <http://www.datastax.com/documentation/cassandra/2.0/cassandra/gettingStartedCassandraIntro.html>
- <http://www.datastax.com/documentation/cql/3.1/pdf/cql31.pdf>
- http://www.datastax.com/documentation/cassandra/2.0/cassandra/dml/dml_config_consistency_c.html

THANK YOU