45. Cassandra – CQL Data Types

Cassandra CQLsh : Cassandra CQLsh stands for Cassandra CQL shell. CQLsh specifies how to use Cassandra commands. After installation, Cassandra provides a prompt Cassandra query language shell (cqlsh). It facilitates users to communicate with it.

Cassandra Query Language (CQL) facilitates developers to communicate with Cassandra. The syntax of Cassandra query language is very similar to SQL.

Cassandra supports different types of data types :

CQL Type	Constants	Description
Ascii	Strings	US-ascii character string
Bigint	Integers	64-bit signed long
Blob	Blobs	Arbitrary bytes in hexadecimal
boolean	Booleans	True or False
counter	Integers	Distributed counter values 64 bit
decimal	Integers, Floats	Variable precision decimal
double	Integers, Floats	64-bit floating point
Float	Integers, Floats	32-bit floating point
Frozen	Tuples, collections, user defined types	stores cassandra types
Inet	Strings	IP address in ipv4 or ipv6 format
Int	Integers	32 bit signed integer
List		Collection of elements
Map		JSON style collection of elements
Set		Collection of elements
Text	strings	UTF-8 encoded strings
timestamp	Integers, Strings	ID generated with date plus time
timeuuid	uuids	Type 1 uuid
Tuple		A group of 2,3 fields
Uuid	Uuids	Standard uuid
varchar	Strings	UTF-8 encoded string
Varint	Integers	Arbitrary precision integer

46. Creating Keyspace, Alter, Drop Keyspace in Cassandra.

Keyspace:

A keyspace is an object that is used to hold column families, user defined types. A keyspace is like RDBMS database which contains column families, indexes, user defined types, data center awareness, strategy used in keyspace, replication factor, etc.

<u>Create Keyspace:</u>

In Cassandra, "**Create Keyspace**" command is used to create keyspace.

Syntax: **CREATE** KEYSPACE <identifier> **WITH** <properties>

Or

CREATE KEYSPACE KeyspaceName

WITH replicaton = { 'class' :strategy **name**,

'replication_factor': **No of** replications **on** different nodes

}

Different components of Cassandra Keyspace

Strategy: There are two types of strategy declaration in Cassandra syntax:

- **Simple Strategy:** Simple strategy is used in the case of one data center. In this strategy, the first replica is placed on the selected node and the remaining nodes are placed in clockwise direction in the ring without considering rack or node location.
- **Network Topology Strategy:** This strategy is used in the case of more than one data centers. In this strategy, you have to provide replication factor for each data center separately.

Replication Factor: Replication factor is the number of replicas of data placed on different nodes. More than two replication factor are good to attain no single point of failure. So, 3 is good replication factor.

Example:

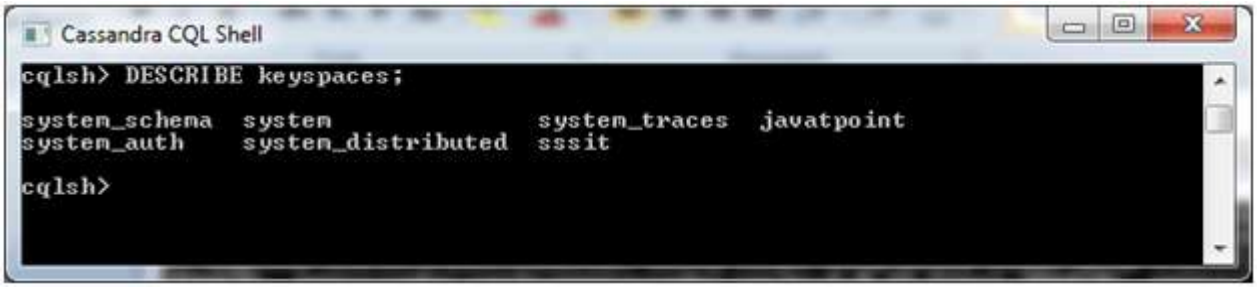
CREATE KEYSPACE College

WITH replication = {'class':'SimpleStrategy', 'replication_factor' : 3};

Output: Keyspace is created

Verification:

To check whether the keyspace is created or not, use the "DESCRIBE" command. By using this command you can see all the keyspaces that are created.



Using a Keyspace : To use the created keyspace, you have to use the USE command.

Syntax: **USE <identifier>**

Ex: USE College

Alter Keyspace : The "ALTER keyspace" command is used to alter the replication factor, strategy name and durable writes properties in created keyspace in Cassandra.

Syntax: **ALTER** KEYSPACE <identifier> **WITH** <properties>

Or

ALTER KEYSPACE KeyspaceName

WITH replicaton = { 'class' :strategy **name**,

'replication_factor': **No of** replications **on** different nodes

}

Ex: ALTER KEYSPACE College

WITH replication = {'class':'NetworkTopologyStrategy', 'replication_factor' : 1};

Drop Keyspace : In Cassandra, "DROP Keyspace" command is used to drop keyspaces with all the data, column families, user defined types and indexes from Cassandra.

Cassandra takes a snapshot of keyspace before dropping the keyspace. If keyspace does not exist in the Cassandra, Cassandra will return an error unless IF EXISTS is used.

Syntax: **DROP** keyspace KeyspaceName ;

Ex: **DROP keyspace College;**

47. Create Table, Alter, Drop, Truncate in Cassandra

Cassandra Create Table

In Cassandra, CREATE TABLE command is used to create a table. Here, column family is used to store data just like table in RDBMS.

So, you can say that CREATE TABLE command is used to create a column family in Cassandra.

Syntax:

CREATE (**TABLE** | COLUMNFAMILY) <tablename>

('<column-definition>' , '<column-definition>')

(**WITH** <**option**> AND <**option**>)

For declaring a primary key:

CREATE TABLE tablename (column1 **name** datatype PRIMARYKEY,

column2 **name** data type, column3 **name** data type.

- - - -)

You can also define a primary key by using the following syntax:

Create table TableName (ColumnName DataType, ColumnName DataType,

- - - ,

Primary key(ColumnName)) **with** PropertyName=PropertyValue;

There are two types of primary keys:

- **Single primary key:** Use the following syntax for single primary key.

  Primary key (ColumnName)

- **Compound primary key:** Use the following syntax for single primary key.

  Primary key(ColumnName1,ColumnName2 . . .)

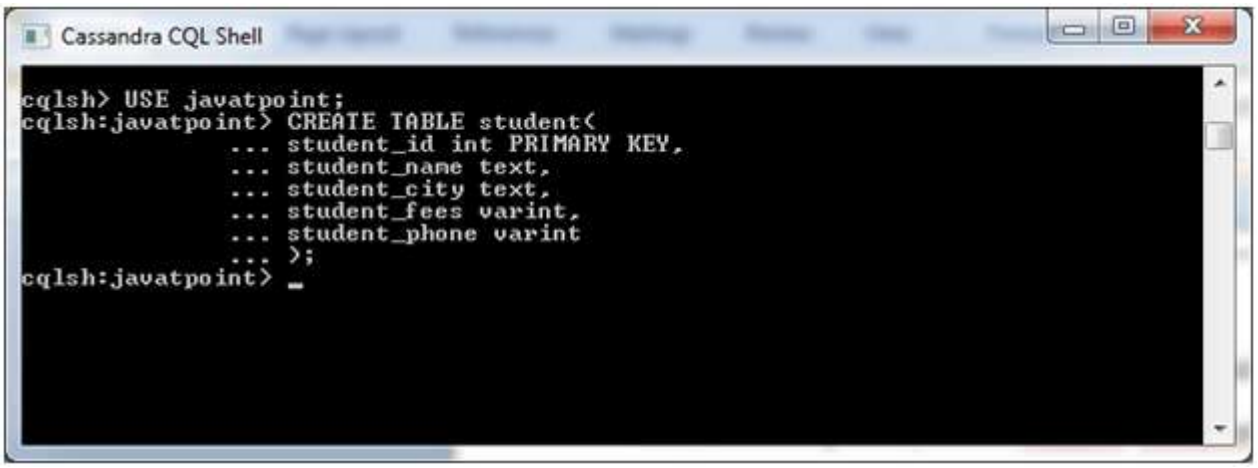
Example:

CREATE TABLE student (student_id **int PRIMARY KEY**, student_name text,

student_city text, student_fees varint,

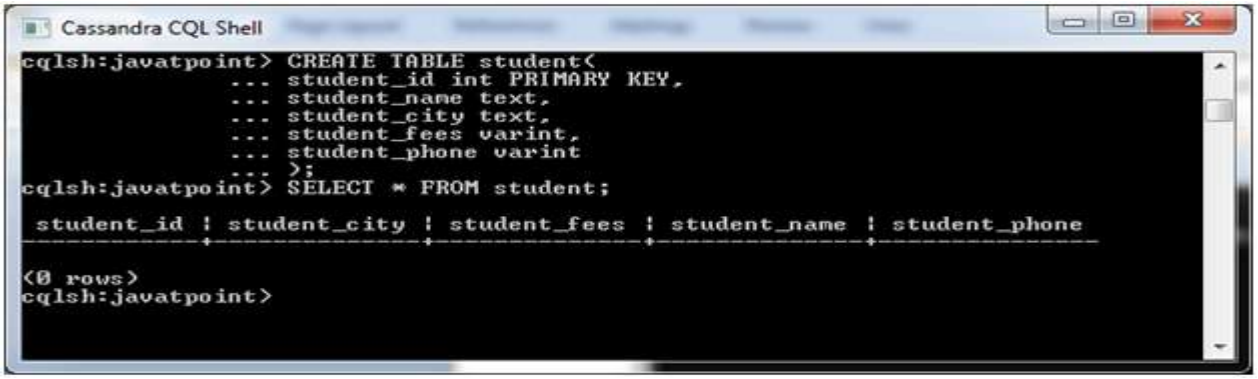
student_phone varint

);



The table is created now. You can check it by using the following command.

SELECT * **FROM** student;



Cassandra Update Data

UPDATE command is used to update data in a Cassandra table. If you see no result after updating the data, it means data is successfully updated otherwise an error will be returned. While updating data in Cassandra table, the following keywords are commonly used:

- **Where:** The WHERE clause is used to select the row that you want to update.
- **Set:** The SET clause is used to set the value.
- **Must:** It is used to include all the columns composing the primary key.

Syntax: **UPDATE** <tablename>

SET <**column name**> = <new value> , <**column name**> = <value>....

WHERE <condition>

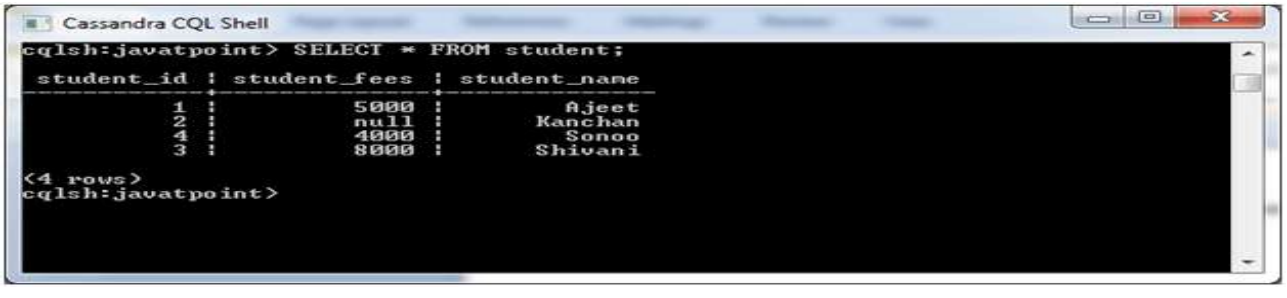
Update KeyspaceName.TableName **Set** ColumnName1=new Column1Value,

ColumnName2=new Column2Value, ColumnName3=new Column3Value,

- - - -

Where ColumnName=ColumnValue

Example:



Here, we update student_fees of student_id 2 to 10000 and student_name to Rahul.

UPDATE student **SET** student_fees=10000,student_name='Rahul'

WHERE student_id=2;

ALTER TABLE

Alter table means, Changes in the datatype of a columns, adding new columns, drop existing columns, renames columns, and change table properties. The command returns no results.

Restriction: Altering PRIMARY KEY columns is not supported. Altering columns in a table that has a materialized view is not supported.

ALTER TABLE [keyspace_name.] table_name

[ALTER column_name TYPE cql_type]

[ADD (column_definition_list)]

[DROP column_list | COMPACT STORAGE]

[RENAME column_name TO column_name]

[WITH table_properties];

Adding a Column

Syntax: **ALTER TABLE table name ADD** new **column** datatype;

Example: **ALTER TABLE** student **ADD** student_email text;

DROP a Column

Syntax: **ALTER TABLE table name DROP column_Name**;

Example: **ALTER TABLE** student **DROP** student_email ;

DROP TABLE

DROP TABLE command is used to drop a table.

Syntax: **DROP TABLE** <tablename>

Example: DROP TABLE student;

Truncate Table

TRUNCATE command is used to truncate a table. If you truncate a table, all the rows of the table are deleted permanently.

Syntax: TRUNCATE <tablename>

Example: TRUNCATE student;

Now, the table is truncated. You can verify it by using SELECT command.

SELECT * FROM student;

48. Perform CRUD (Create, Read, Update, and Delete) operations.

Answer: Write some examples from 47,49,50,51 Questions.