# Cassandra

## Part I setting up Cassandra environment

1. Set up Java environment (Cassandra environment is based on the jdk8)

* Go the main page of oracle <https://www.oracle.com/index.html>
* Find “Developers”

A screenshot of a cell phone

Description automatically generated

* Find “Java”

A picture containing monitor, screen

Description automatically generated

* Click “Java SE Download”

A screenshot of a cell phone

Description automatically generated

* Choose Java SE8 and close download, because Cassandra may not work with JDK9 and newer release version.

A screenshot of a cell phone

Description automatically generated

* Download jdk8

A screenshot of a cell phone

Description automatically generated

* Choose “Accept License Agreement” before download, and choose the matched operating system file to download

A screenshot of a cell phone

Description automatically generated

* After downloading, you can install according to the instruction.
* Type command “java -version” in terminal to check the install of jdk

1. Set up Cassandra environments

* Go to the wibesite <http://cassandra.apache.org/download/> and choose the latest version

A screenshot of a social media post

Description automatically generated

* Download the bin.tar.gz file
* Type command in terminal to set up Cassandra environment

tar -zxvf apache-cassandra-3.11.5-bin.tar.gz

mkdir Cassandra

mv apache-cassandra-3.11.5/\* Cassandra

## Part 2 start with Cassandra

1. Enter the Cassandra file, run the server with command

./bin/cassandra -f

A screenshot of a newspaper

Description automatically generated

1. Then open another terminal, run cqlshell command to connect the server

./bin/cqlsh

./bin/cqlsh localhost 9042 # connect to a specified server

A screenshot of a cell phone

Description automatically generated

## Part 3 start with Keyspace

1. Check the keyspace list

DESCRIBE KEYSPACES;

1. Create keyspace

CREATE KEYSPACE <name>

WITH REPLICATION = {'class':'Strategy Name','replication\_factor': int}

AND durable\_writes = boolean;

e.g.

CREATE KEYSPACE test WITH replication = {'class': 'SimpleStrategy', 'replication\_factor' : 3};

CREATE KEYSPACE test2 WITH replication = {'class': 'NetworkTopologyStrategy', 'DC1' : 1, 'DC2' : 3} AND durable\_writes = true;

Strategy class

Simple strategy: simple strategy, used in the case of a data center

Network topology strategy: network topology strategy for multiple data centers

Replication factor: number of replicas

Persistent write attribute: durable\_write: Boolean value, default true

1. Update keyspace

ALTER KEYSPACE <name>

WITH REPLICATION = {'class': 'strategy name', 'replication\_factor': int}

AND durable\_writes = boolean;

1. Use keyspace

USE <keyspace\_name>;

1. Drop keyspace

DROP KEYSPACE <name>;

## Part 4 start with table

1. Check info of all tables

DESCRIBE TABLES;

1. Create table

CREATE TABLE [IF NOT EXISTS] <name> (

column cql\_type,

column cql\_type,

column cql\_type,

RIMARY KEY(column, column)

) [WITH property = value AND property = value ];

e.g:

CREATE TABLE users (

id bigint primary key,

username text,

age int,

height double,

brithday date,

isvip boolean,

salt uuid,

ip inet,

hobbies list<text>,

skills set<text>,

scores map<text, int>,

tags tuple<text, text>,

createtime timestamp

) ;

1. Check info of table

DESCRIBE TABLE name;

1. Update table

* Add a column

ALTER TABLE name ADD column cql\_type;

e.g.

ALTER TABLE users ADD temp varchar;

* Delete a column

ALTER TABLE <name> DROP column

e.g.

ALTER TABLE users DROP temp;

* Delete columns

ALTER TABLE <name> DROP (column, column)

1. Drop table

DROP TABLE name;

1. Truncate table: delete all rows in the table

truncate <name>;

## Part 5 start with Index

1. Create Index

CREATE INDEX [name] ON <table\_name(column)>

e.g.

CREATE INDEX users\_username\_idx ON users(username);

1. Delete Index

DROP INDEX [IF EXISTS] <name>;

## Part 6 start with data

1. Insert data

INSERT INTO <name>(column, column)

VALUES(value, value)

[USING TTL seconds];

or

INSERT INTO <name>

JSON ' {"key1": "value", "key2": "value"} ';

e.g.

INSERT INTO users(

id,

username,

age,

height,

birthday,

isvip,

salt,

ip,

hobbies,

skills,

scores,

tags,

createtime)

VALUES(

1,

'mengday',

26,

135.5,

'1990-10-26',

true,

uuid(),

'192.168.1.1',

['java', 'iOS'],

{'eat', 'drink'},

{'china': 80, 'english': 90},

('mm', 'money'),

dateof(now())

);

INSERT INTO users(

id,

username,

age,

height,

birthday,

isvip,

salt,

ip,

hobbies,

skills,

scores,

tags,

createtime)

VALUES(

2,

'mengdee',

36,

145.5,

'1989-06-06',

false,

blobAsUuid(timeuuidAsBlob(now())),

'192.168.11.11',

['java', 'php'],

{'play', 'happy'},

{'china': 90, 'english': 99},

('gg', 'rmb'),

dateof(now())

);

INSERT INTO users

JSON ' {"id": 3, "username": "mengdie", "age": 16}'

using ttl 3600;

1. Select data

SELECT column, column FROM<name> WHERE <condition>;

e.g.

SELECT \* FROM users;

// If the field used in the where condition does not create an index, you need to use allow filtering to force the query.

SELECT id, username, createtime, tags FROM users WHERE id in(1, 2) and age > 18 and tags = ('mm', 'money') allow filtering;

1. Update data

UPDATE <tableName> [USING TTL seconds]

SET

column = value,

column = value

WHERE <condition>

e.g.

UPDATE users using ttl 60 SET username = 'hehe' WHERE id = 3;

// When the updated condition is not met, it is equivalent to the insert operation.

UPDATE users SET username = 'admin' WHERE id = 999999;

1. Delete data

* Delete rows

DELETE FROM <name> WHERE <condition>;

* Delete columns

DELETE column FROM <name> WHERE <condition>;

1. Batch operation

begin batch

<insert-stmt>;

<update-stmt>;

<delete-stmt>;

apply batch;

e.g.

begin batch

insert into users json ' {"id": 4, "username": "test", "age": 16}';

update users set age = 20 where id = 4;

delete age from users where id = 4;

apply batch;

1. UDT(User designed type)

* Create

CREATE TYPE <NAME> (

column cql\_type,

column cql\_type

);

CREATE TYPE address (

proivnce text,

city text,

region text,

town text

);

* // List all types

describe types;

* // check specific type

describe type address;

* // add column in type

ALTER TYPE <name> ADD column cql\_type;

alter type address add way text;

* // rename the column in type

ALTER TYPE <name> RENAME <COLUMN> TO <new\_name>

alter type address rename way to road;

* // delete type

DROP TYPE <name>;

drop type address;

## Part 7 start with Collection operation

1. Set

* Create with set<cql\_type>,
* Use with {value, value}
* // add element

UPDATE users SET skills = skills + {'eat', 'drink', 'mm'} WHERE id = 2;

* // delete element

UPDATE users SET skills = skills - {'eat', 'mm'} WHERE id = 2;

1. List

* Create with list<cql\_type>,
* Use with [value, value]
* //add element

UPDATE users SET hobbies = hobbies + ['php', 'javascript'] WHERE id = 1;

UPDATE users SET hobbies = ['go'] + hobbies WHERE id = 1;

* //delete element

UPDATE users SET hobbies = hobbies - ['php', 'javascript'] WHERE id = 1;

* // update element in specific index

UPDATE users SET hobbies[0] = 'golang' WHERE id = 1;

* // delete element in specific index

DELETE hobbies[0] FROM users WHERE id = 1;

1. Map

* Create with <cql\_type, cql\_type>
* Use with {'key':value, 'key':value}
* // add element

UPDATE users SET scores = scores + {'math': 80, 'physics': 88} WHERE id = 1;

* // delete element

UPDATE users SET scores = scores - {'math', 'physics'} WHERE id = 1;

DELETE scores['english'] FROM users WHERE id = 1;

* // update element

UPDATE users SET scores['china'] = 100 WHERE id = 1;

1. Tuple

* Create with tuple<cql\_type, ..., cql\_type>
* Use with (value, ...., value)
* // update element

UPDATE users SET tags = ('girl', '$') WHERE id = 1;

1. Collection data filtering by contains

* // list、set

SELECT \* FROM users WHERE hobbies CONTAINS 'php' allow filtering;

* // map

SELECT \* FROM users WHERE scores CONTAINS KEY 'english' allow filtering;

## Part 8 start with Materialized View

1. Create view

CREATE MATERIALIZED VIEW [IF NOT EXISTS] <name> AS

select\_statement

PRIMARY KEY (column, column)

[with table\_options];

e.g.

CREATE MATERIALIZED VIEW user\_view AS

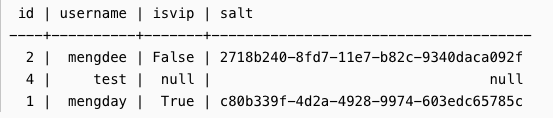
select id, username, salt, isvip from users where username is not null

PRIMARY KEY (id, username)

with comment = 'users view';

1. Check the view

SELECT \* FROM user\_view;

 1

1. Alter view options

ALTER MATERIALIZED VIEW <name> WITH table\_options

1. Delete view

DROP MATERIALIZED VIEW [IF EXISTS] <name>

## Part 9 start with functions

1. Predefined function

* count()：Number of rows count(\*)
* now(): current time
* uuid(): Generate a uuid value
* min()：Find the minimum
* max()：Find the maximum
* sum()：Summation
* avg()：Average
* cast(column as cql\_type): Convert to other basic data types
* minTimeuuid(): minTimeuuid(‘2013-02-02 10:00+0000’)
* maxTimeuuid(): maxTimeuuid(‘2013-01-01 00:05+0000’)

1. (timeuuid、date、timestamp、bigInt) Mutual conversion function

|  |  |  |
| --- | --- | --- |
| Function name | Input type | Description |
| toDate | timeuuid | Converts the timeuuid argument into a date type |
| toDate | timestamp | Converts the timestamp argument into a date type |
| toTimestamp | timeuuid | Converts the timeuuid argument into a timestamp type |
| toTimestamp | date | Converts the date argument into a timestamp type |
| toUnixTimestamp | timeuuid | Converts the timeuuid argument into a bigInt raw value |
| toUnixTimestamp | timestamp | Converts the timestamp argument into a bigInt raw value |
| toUnixTimestamp | date | Converts the date argument into a bigInt raw value |
| dateOf | timeuuid | Similar to toTimestamp(timeuuid) (DEPRECATED) |
| unixTimestampOf | timeuuid | Similar to toUnixTimestamp(timeuuid) (DEPRECATED) |

e.g.

SELECT cast(height as int) FROM users;

SELECT count(\*) as count, min(height) as min, max(height) as max, sum(height) as sum, avg(height) as avg, now() as now, uuid() as uuid FROM users;

1. Custom function

create [or replace] function [if not exists] <function\_name>(arg1 int, arg2 text, ...)

returns null on null input

returns <cql\_type>

language java

as $$

// some java code

return arg;

$$;

or

create function if not exists <keyspace>.<function\_name>(argname cql\_type)

called on null input

returns <cql\_type>

language java

as $$

// some java code

$$;