

Cassandra

1. Развернуть касандру с репы docker-cassandra с ветки bde-cassandra (или на учебном кластере просто подключиться)
2. Подключить к Cassandra
3. Создать таблицы
4. Вставить записи
5. Изучить особенности работы where
6. Создать таблицу с несколькими primary key, вставить значения, пофилтровать по ним (как по 1му, так и по 2м)
7. Изучить особенности хранения данных
8. Сделать то же самое с hbase

1. Развернул Cassandra с репы docker-cassandra с ветки bde-cassandra [docker-cassandra](#)
Скопировал с репы папку.
Разжал, перешел, запустил контейнер, запустил Cassandra

docker-compose -f docker-compose-cluster.yml up -d

```
max@max:~/Downloads/docker-cassandra-bde-cassandra$ docker-compose -f docker-com
pose-cluster.yml up -d
Creating network "docker-cassandra-bde-cassandra_default" with the default drive
r
Pulling cassandra-1 (bde2020/cassandra:)...
latest: Pulling from bde2020/cassandra
```

2. Подключить к Cassandra

docker exec -it cassandra-1 bash

```
Creating cassandra-1 ... done
Creating cassandra-2 ... done
max@max:~/Downloads/docker-cassandra-bde-cassandra$ docker exec -it cassandra-1
bash
root@899865da893e:/# cqlsh
```

3. Создать таблицы

CREATE TABLE if not exists lesson7.aircrafts_new (
 ... id int,
 ... name_craft text,
 ... type_craft text,
 ... year date,
 ... country text,
 ... primary key (id));

4. Вставить записи

insert into aircrafts_new (id, name_craft, type_craft, year, country)
values (1, 'Mig-3', 'F', 1939, 'USSR');

**insert into aircrafts_new (id, name_craft, type_craft, year, country)
values (2, 'Yak-1b', 'F', 1937, 'USSR');**

```
cqlsh:lesson7>
cqlsh:lesson7> insert into aircrafts_new (id, name_craft, type_craft, year, country)
... values (5, 'He-157', 'A', 1933, 'GER');
cqlsh:lesson7>
cqlsh:lesson7> select * from aircrafts_new;
```

id	country	name_craft	type_craft	year
5	GER	He-157	A	-2147481715
1	USSR	Mig-3	F	-2147481709
2	USSR	Yak-1b	F	-2147481711
4	GER	Bf-109E	F	-2147481712
3	USSR	TB-3	B	-2147481714

(5 rows)
cqlsh:lesson7>

5. Изучить особенности работы where
Можем обратиться к данным только по id (под это она и «заточена»)
6. Создать таблицу с несколькими primary key, вставить значения, профильтровать по ним (как по 1му, так и по 2м)

```
id | type_craft | country | name_craft | year
---+---+---+---+---
5 | A | GER | Ju-87 | 1933
5 | B | GER | Ju-88 | 1933
1 | F | USSR | Mig-3 | 1939
2 | F | USSR | Yak-1b | 1937
4 | F | GER | Bf-109E | 1936
3 | B | USSR | TB-3 | 1934
(6 rows)
cqlsh:lesson7> select * from aircrafts_2pk where id=5
... ;
```

id	type_craft	country	name_craft	year
5	A	GER	Ju-87	1933
5	B	GER	Ju-88	1933

7. Изучить особенности хранения данных
8. Сдлать то же самое с hbase

Hbase

Скачал с репы: <https://github.com/big-data-europe/docker-hbase>

1. Развернул hbase с репы <https://github.com/big-data-europe/docker-hbase> (или на учебном кластере просто подключиться – слишком просто через MobaXterm_Personal_21.5.exe)
2. Подключить к hbase

docker-compose -f docker-compose-standalone.yml up -d

```
max@max:~/Downloads/docker-hbase-master$ docker-compose -f docker-compose-standalone.yml up -d
Creating network "docker-hbase-master_default" with the default driver
Creating volume "docker-hbase-master_hadoop_namenode" with default driver
Creating volume "docker-hbase-master_hadoop_datanode" with default driver
Creating volume "docker-hbase-master_hadoop_historyserver" with default driver
```

Далее как-то криво запускался Hbase, но запустился:

docker exec -it e5ba25121ea1 bash

```
max@max:~/Downloads/docker-hbase-master$ docker exec -it e5ba25121ea1 bash
root@e5ba25121ea1:/# hbase shell
bash: hbase: command not found
root@e5ba25121ea1:/# hbase shell
2021-12-28 11:53:39,151 WARN [main] util.NativeCodeLoader: Unable to load native-hadoop
sses where applicable
2021-12-28 11:53:56,041 ERROR [main] zookeeper.RecoverableZooKeeper: ZooKeeper exists fai
```

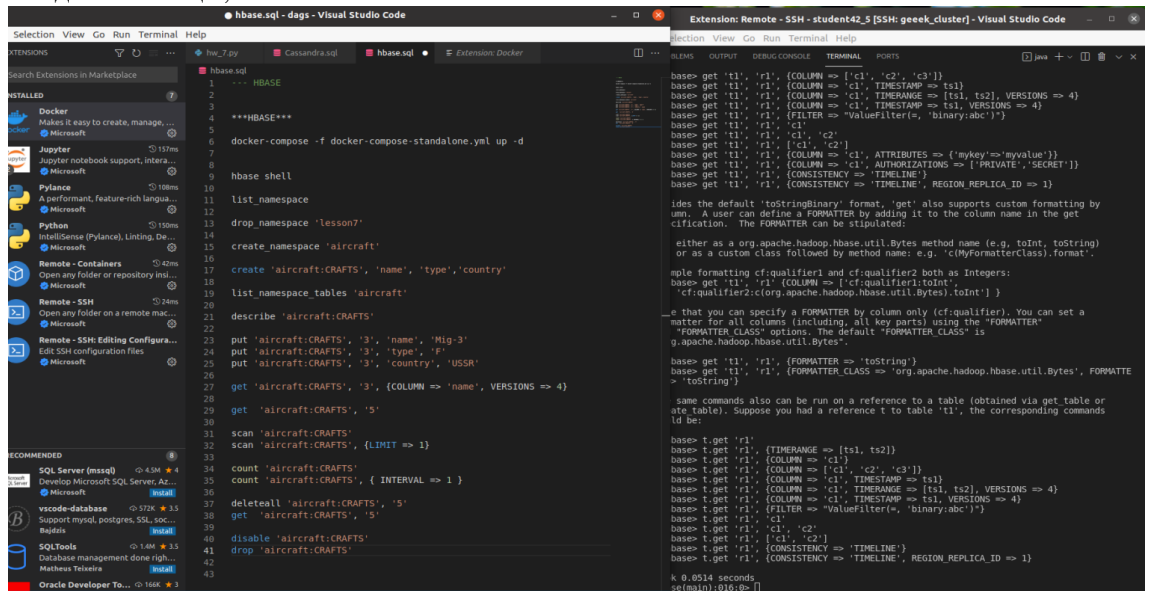
И опять слетел.

Поэтому заранился через VSCode к учебному кластеру:

Настройка подключения через VSCode (Урок_5 29:50)

Статья: <https://dker.ru/docs/vscode-docker/>

3. Создать таблицы, Вставить записи



```
hbase.sql - dags - Visual Studio Code
1 --- HBASE
2
3 ***HBASE***
4
5 docker-compose -f docker-compose-standalone.yml up -d
6
7 hbase shell
8
9 list_namespace
10
11 drop_namespace 'lesson7'
12
13 create_namespace 'aircraft'
14
15 list_namespace_tables 'aircraft'
16
17 describe 'aircraft:CRAFTS'
18
19 put 'aircraft:CRAFTS', '3', 'name', 'Mig-3'
20
21 put 'aircraft:CRAFTS', '3', 'type', 'F'
22
23 put 'aircraft:CRAFTS', '3', 'country', 'USSR'
24
25 get 'aircraft:CRAFTS', '3', {COLUMN => 'name', VERSIONS => 4}
26
27 get 'aircraft:CRAFTS', '5'
28
29 scan 'aircraft:CRAFTS'
30
31 scan 'aircraft:CRAFTS', {LIMIT => 1}
32
33 count 'aircraft:CRAFTS'
34
35 count 'aircraft:CRAFTS', {INTERVAL => 1}
36
37 deleteall 'aircraft:CRAFTS', '5'
38
39 get 'aircraft:CRAFTS', '5'
40
41 disable 'aircraft:CRAFTS'
42
43 drop 'aircraft:CRAFTS'

Extension: Remote - SSH - student42_5 [SSH: geek_cluster] - Visual Studio Code
base> get 't1', 'r1', {COLUMN => ['c1', 'c2', 'c3']}
base> get 't1', 'r1', {COLUMN => 'c1', TIMESTAMP => ts1}
base> get 't1', 'r1', {COLUMN => 'c1', TIMERANGE => {ts1, ts2}, VERSIONS => 4}
base> get 't1', 'r1', {COLUMN => 'c1', TIMESTAMP => ts1, VERSIONS => 4}
base> get 't1', 'r1', {FILTER => "ValueFilter(=, 'binary:abc')"}
base> get 't1', 'r1', 'c1'
base> get 't1', 'r1', 'c2', 'c3'
base> get 't1', 'r1', ['c1', 'c2']
base> get 't1', 'r1', {COLUMN => 'c1', ATTRIBUTES => {'mykey'=>'myvalue'}}
base> get 't1', 'r1', {COLUMN => 'c1', AUTHORIZATIONS => {'PRIVATE', 'SECRET'}}
base> get 't1', 'r1', {CONSISTENCY => 'TIMELINE'}
base> get 't1', 'r1', {CONSISTENCY => 'TIMELINE', REGION_REPLICA_ID => 1}

uses the default 'toStringBinary' format, 'get' also supports custom formatting by
user. A user can define a FORMATTER by adding it to the column name in the get
configuration. The FORMATTER can be stipulated:

either as a org.apache.hadoop.hbase.util.Bytes method name (e.g. toInt, toString)
or as a custom class followed by method name: e.g. 'c(MyFormatterClass).format'.

example formatting cf:qualifier1 and cf:qualifier2 both as Integers:
base> get 't1', 'r1' {COLUMN => {cf:qualifier1:toInt,
cf:qualifier2:c(org.apache.hadoop.hbase.util.Bytes).toInt}}

e that you can specify a FORMATTER by column only (cf:qualifier). You can set a
formatter for all columns (including, all key parts) using the "FORMATTER"
"FORMATTER CLASS" options. The default "FORMATTER CLASS" is
org.apache.hadoop.hbase.util.Bytes.

base> get 't1', 'r1', {FORMATTER => 'toString'}
base> get 't1', 'r1', {FORMATTER_CLASS => 'org.apache.hadoop.hbase.util.Bytes', FORMATTER
> 'toString'}

same commands also can be run on a reference to a table (obtained via get table or
ate table). Suppose you had a reference t to table 't1', the corresponding commands
ld be:

base> t.get 'r1'
base> t.get 'r1', {TIMERANGE => {ts1, ts2}}
base> t.get 'r1', {COLUMN => 'c1'}
base> t.get 'r1', {COLUMN => ['c1', 'c2', 'c3']}
base> t.get 'r1', {COLUMN => 'c1', TIMESTAMP => ts1}
base> t.get 'r1', {COLUMN => 'c1', TIMERANGE => {ts1, ts2}, VERSIONS => 4}
base> t.get 'r1', {COLUMN => 'c1', TIMESTAMP => ts1, VERSIONS => 4}
base> t.get 'r1', {FILTER => "ValueFilter(=, 'binary:abc')"}
base> t.get 'r1', 'c1'
base> t.get 'r1', 'c2', 'c3'
base> t.get 'r1', ['c1', 'c2']
base> t.get 'r1', {CONSISTENCY => 'TIMELINE'}
base> t.get 'r1', {CONSISTENCY => 'TIMELINE', REGION_REPLICA_ID => 1}

K 0.0514 seconds
se(main):016:0x []
```