Cassandra

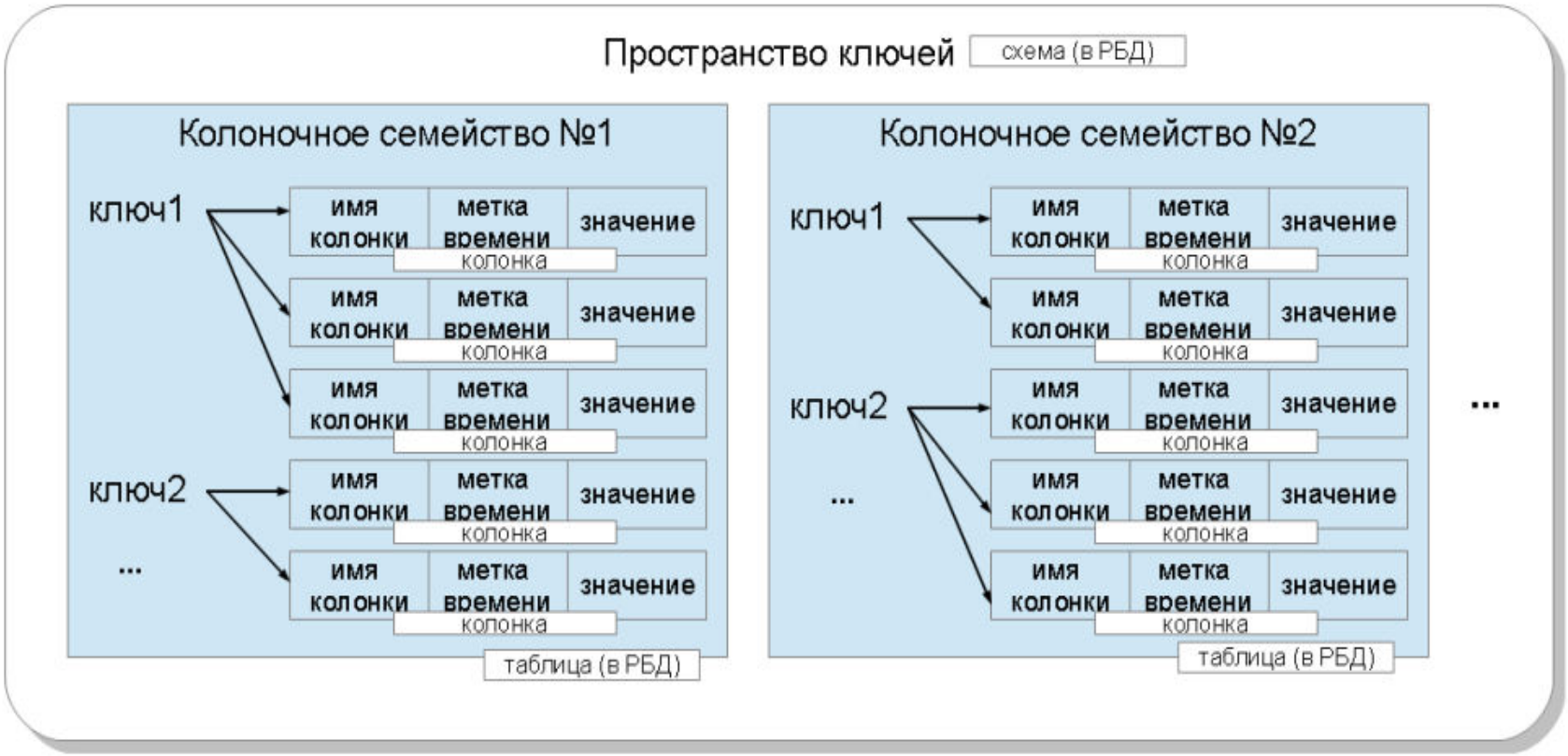
Выполнили:
студенты группы БПЗ 1101
Васина А.В.
Сидоров А.С.

Cassandra

- Cassandra — децентрализованная, отказоустойчивая и надёжная база данных “ключ-значение”.

Терминология

- Пространство ключей (keyspace) - понятие схемы базы данных (database schema) в реляционной модели
- Колоночные семейства (column family) - реляционная таблица
- Колонки (column) - объединяются при помощи ключа (row key) в записи (row)
- Колонка состоит из трех частей: имени (column name), метки времени (timestamp) и значения (value).
- Колонки в пределах записи упорядочены.
- В отличие от реляционной БД, никаких ограничений на то, чтобы записи (а в терминах БД это строки) содержали колонки с такими же именами как и в других записях — нет.


Пространство ключей
схема (в РБД)
Колоночное семейство №1
ключ1
имя колонки
метка времени
значение
колонка
имя колонки
метка времени
значение
колонка
имя колонки
метка времени
значение
колонка
ключ2
имя колонки
метка времени
значение
колонка
...
имя колонки
метка времени
значение
колонка
таблица (в РБД)
Колоночное семейство №2
ключ1
имя колонки
метка времени
значение
колонка
имя колонки
метка времени
значение
колонка
ключ2
имя колонки
метка времени
значение
колонка
...
имя колонки
метка времени
значение
колонка
имя колонки
метка времени
значение
колонка
таблица (в РБД)
...

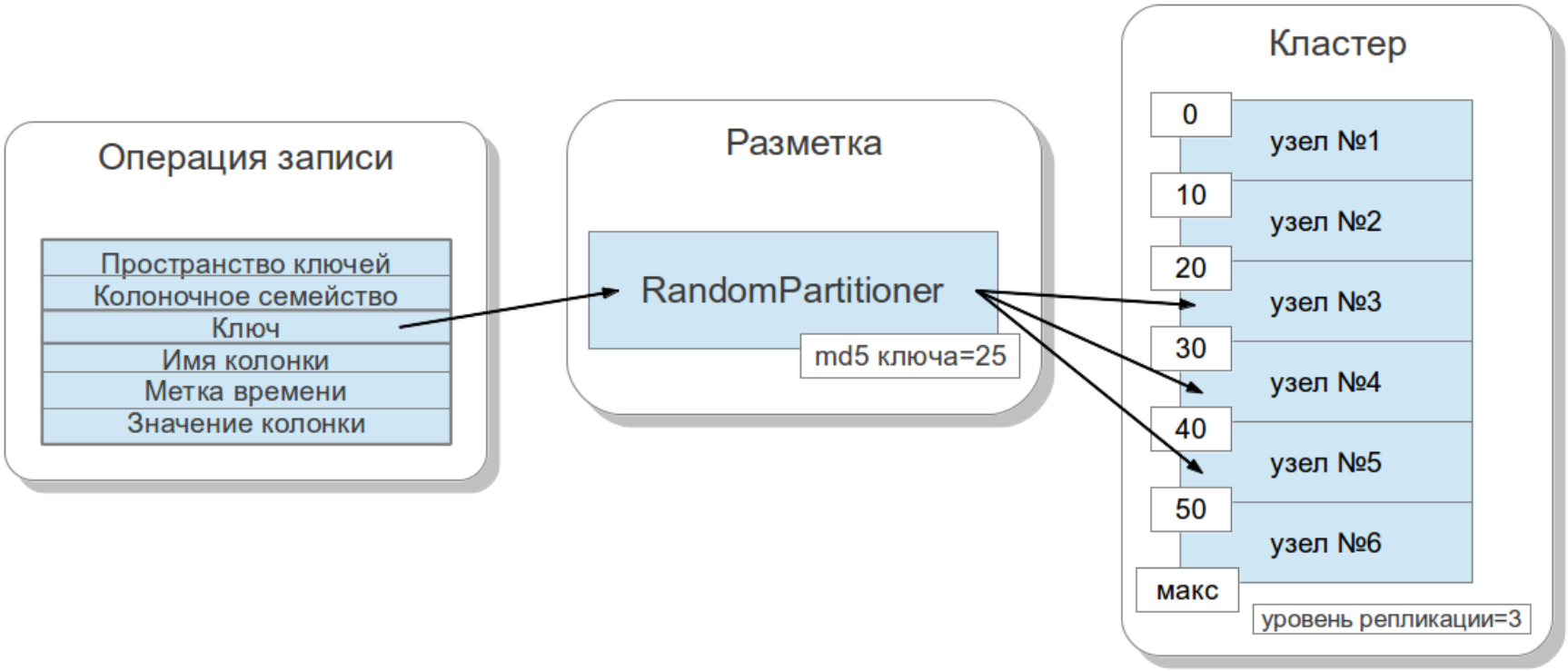
- Конкретное значение, хранимое в кассандре идентифицируется:
- пространством ключей — это привязка к приложению (предметной области). Позволяет на одном кластере размещать данные разных приложений;
- колоночным семейством — это привязка к запросу;
- ключом — это привязка к узлу кластера. От ключа зависит на какие узлы попадут сохранённые колонки;
- именем колонки — это привязка к атрибуту в записи. Позволяет в одной записи хранить несколько значений.
- С каждым значением связана метка времени — задаваемое пользователем число, которое используется для разрешения конфликтов во время записи: чем больше число, тем колонка считается новее, а при сравнении перетирает старые колонки.
- В кассандре все операции записи данных это всегда операции перезаписи

Типы данных

- BytesType: любые байтовые строки (без валидации)
- AsciiType: ASCII строка
- UTF8Type: UTF-8 строка
- IntegerType: число с произвольным размером
- Int32Type: 4-байтовое число
- LongType: 8-байтовое число
- UUIDType: UUID 1-ого или 4-ого типа
- TimeUUIDType: UUID 1-ого типа
- DateType: 8-байтовое значение метки времени
- BooleanType: два значения: true = 1 или false = 0
- FloatType: 4-байтовое число с плавающей запятой
- DoubleType: 8-байтовое число с плавающей запятой
- DecimalType: число с произвольным размером и плавающей запятой
- CounterColumnType: 8-байтовый счётчик

Распределение данных

- Для распределения данных кассандра использует технику, известную как *согласованное хеширование* (consistent hashing).
- Каждому узлу ставится в соответствие *метка* (token)
- Общее число выбранных узлов должно быть равным уровню репликации (replication factor).
- Уровень репликации задаётся для каждого пространства ключей и позволяет регулировать избыточность данных (data redundancy).
- Перед тем, как добавить узел в кластер, необходимо задать ему метку.
- Весь набор меток для кластера называется кольцом (ring).


Операция записи
Пространство ключей
Колоночное семейство
Ключ
Имя колонки
Метка времени
Значение колонки
Разметка
RandomPartitioner
md5 ключа=25
Кластер
0
узел №1
10
узел №2
20
узел №3
30
узел №4
40
узел №5
50
узел №6
макс
уровень репликации=3

Согласованность данных

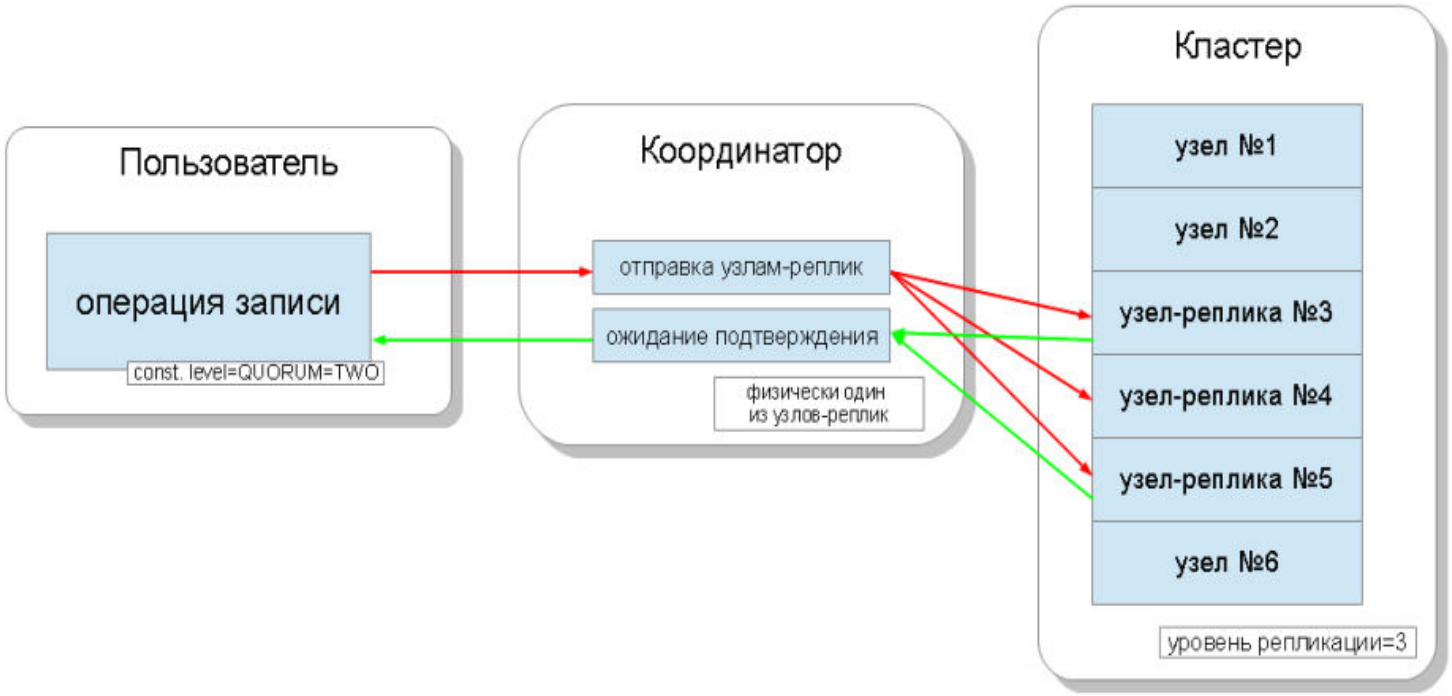
- Для каждого запроса, как на чтение, так и на запись, есть возможность задать уровень согласованности данных.

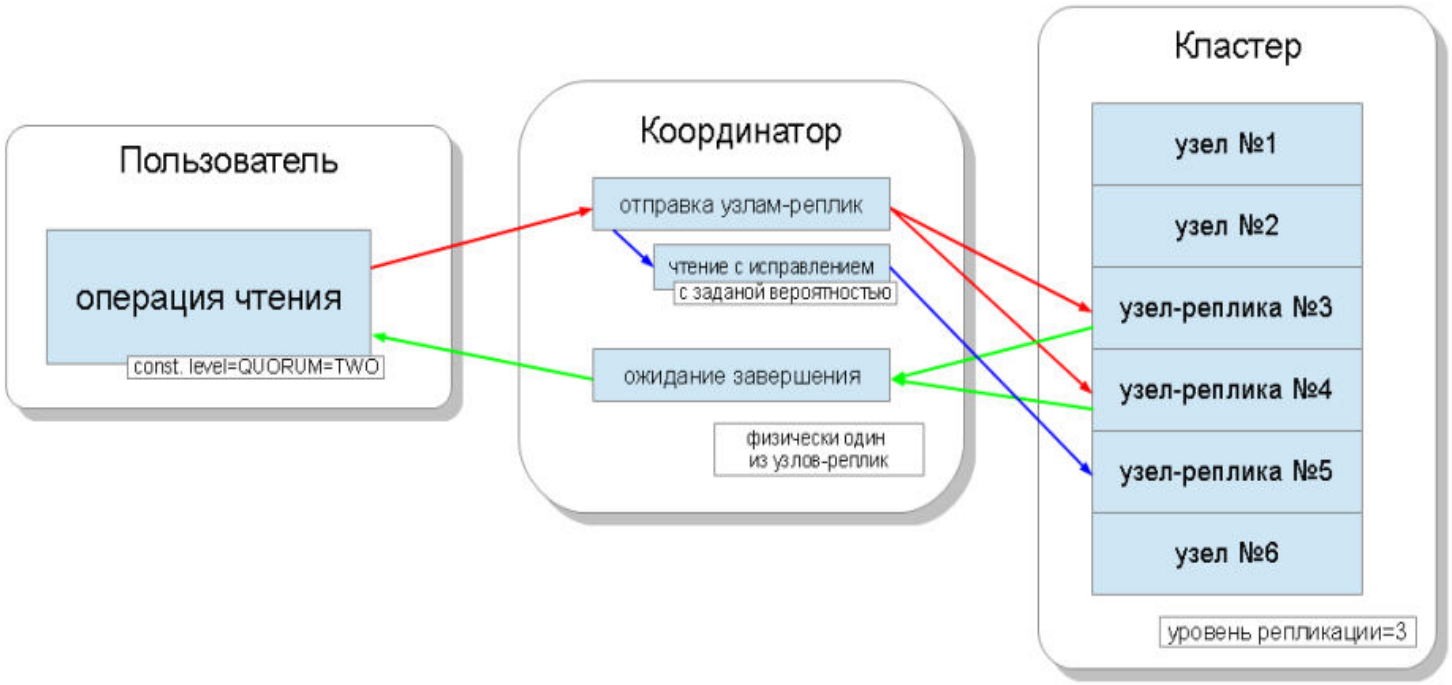
Для записи:

- ONE;
- TWO
- THREE
- QUORUM
- LOCAL_QUORUM
- EACH_QUORUM
- ALL
- ANY

Для чтения:

- ONE
- TWO
- THREE
- QUORUM
- LOCAL_QUORUM
- EACH_QUORUM
- ALL


Пользователь
операция записи
const. level=QUORUM=TWO
Координатор
отправка узлам-реплик
ожидание подтверждения
физически один
из узлов-реплик
Кластер
узел №1
узел №2
узел-реплика №3
узел-реплика №4
узел-реплика №5
узел №6
уровень репликации=3


Пользователь
операция чтения
const. level=QUORUM=TWO
Координатор
отправка узлам-реплик
чтение с исправлением
с заданой вероятностью
ожидание завершения
физически один
из узлов-реплик
Кластер
узел №1
узел №2
узел-реплика №3
узел-реплика №4
узел-реплика №5
узел №6
уровень репликации=3

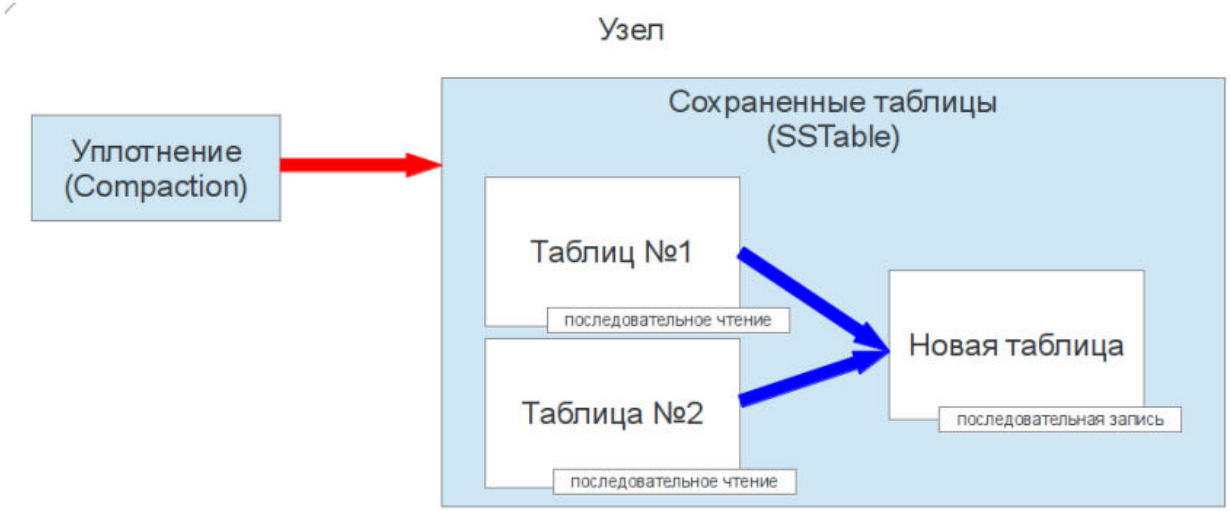
Восстановление данных

Три механизма восстановления данных:

- *чтение с восстановлением* (read repair) — во время чтения данные запрашиваются со всех реплик и сравниваются уже после завершения координации.
- *направленной отправки* (hinted handoff) — позволяет сохранить информацию об операции записи на координаторе в том случае, если запись на какой-либо из узлов не удалась.
- *анти-энтропийное восстановление узла* (anti-entropy node repair) — это некий процесс восстановления всех реплик, который должен запускаться регулярно вручную при помощи команды “nodetool repair” и позволяет поддержать количество реплик всех данных, которые возможно были не восстановлены первыми двумя способами, на требуемом уровне репликации.

Уплотнение

- Чтение последовательно несколько сохранённых таблиц и запись в новую сохранённую таблицу, в которой объединены данные по меткам времени.



Операции удаления

- Операции удаление колонок — это операции записи специального значения — *затирающего значения* (tombstone)

Преимущества

- Очень быстрый процесс записи.

Операция записи так устроена, что она действительно происходит быстро. Cassandra – одно из немногих хранилищ, где запись происходит быстрее чтения.

- Легкость администрирования.
- В HBase: 4 демона на каждый сервер.

Недостатки

- Cassandra не умеет передавать данные поточно
- Слишком много настроек вынесено на cluster-level
- Compaction существенно замедляет запросы
- Протокол общения между нодами: Thrift

Недостатки

- Плохо реализован range scan: кассандра не умеет передавать данные поточно.

Установка Cassandra

Установка напрямую из ASF репозиториев:

- В файл /etc/apt/sources.list добавить:

deb http://www.apache.org/dist/cassandra/debian 11x main

deb-src http://www.apache.org/dist/cassandra/debian 11x main

Для этого вводим команды:

- **su root** – для входа в режим суперпользователя
- **mc** – файловый менеджер
/etc/apt/sources.list
- Находим файл /etc/apt/sources.list
- Жмем **f4**, чтобы изменить его
- Вставляем нужные строки
- **f2, y** – сохранение изменений

Выполняем следующие команды:

- Добавляем PUBLIC_KEY

gpg --keyserver pgp.mit.edu --recv-keys F758CE318D77295D
gpg --export --armor F758CE318D77295D | sudo apt-key add -

- Добавление открытого ключа (начиная с debian packet 0.7.5)

gpg --keyserver pgp.mit.edu --recv-keys 2B5C1B00
gpg --export --armor 2B5C1B00 | sudo apt-key add –

- Установка Cassandra

apt-get update
apt-get install cassandra
service cassandra restart

Начало работы. CQLSH

- **cqlsh** — коммандная строка для C* CQL.
- **CQL** — SQL-подобный язык запросов. Аббревиатура от **C**assandra **Q**uery **L**anguage.
- Запустим cqlsh.

```
root@mtucivb-VirtualBox:/etc/apt# cqlsh
Connected to Test Cluster at localhost:9160.
[cqlsh 2.2.0 | Cassandra 1.1.12 | CQL spec 2.0.0 | Thrift protocol 19.33.0]
Use HELP for help.
```

Создадим keyspace (базу данных).

CREATE KEYSPACE twissandra
WITH strategy_class ='SimpleStrategy‘
AND strategy_options: replication_factor = '1';

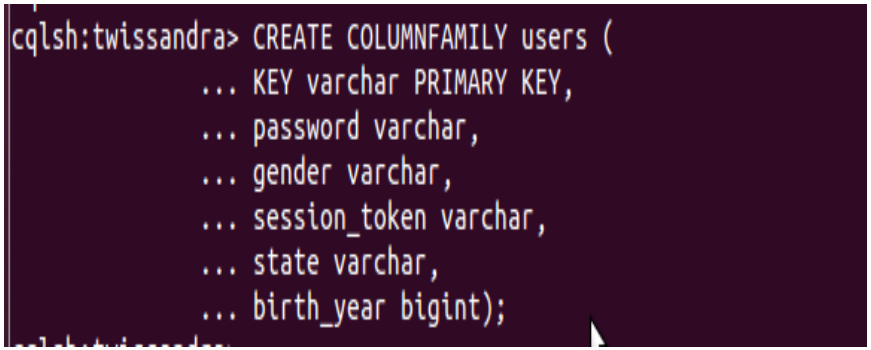
```
cqlsh>
cqlsh> CREATE KEYSPACE twissandra WITH strategy_class = 'SimpleStrategy' AND str
ategy_options:replication_factor = '1';
cqlsh> USE twissandra;
```

Создание Column Family (таблицы).

- Создадим таблицу (которые на самом деле называются Column Family, но для простоты перехода с SQL на CQL используют часто слово table) на CQL и заполним данными.

USE twissandra;

CREATE COLUMNFAMILY users (
KEY varchar PRIMARY KEY,
password varchar,
gender varchar,
session_token varchar,
state varchar,
birth_year bigint);

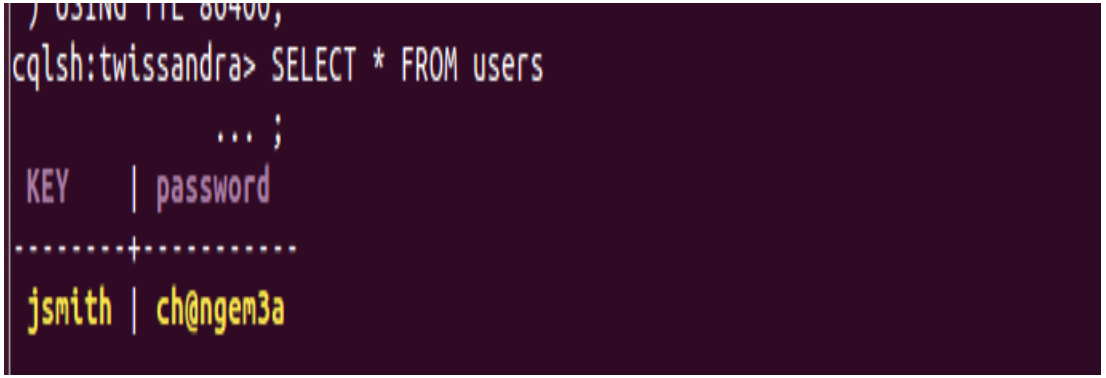


- Таблицы в C* обязаны иметь PRIMARY KEY. Он используется для поиска ноды, в которой хранится искомая строка.

Заполнение данными и чтение данных

INSERT INTO users (KEY, password) VALUES ('jsmith', 'ch@ngem3a') USING TTL 86400;

*SELECT * FROM users;*



Добавление новых колонок

- ALTER COLUMNFAMILY команда позволяет добавлять новые колонки в таблицу.

- Например, чтобы добавить столбец coupon_code типа varchar:

ALTER TABLE users ADD coupon_code varchar;

- Это создает новый столбец и добавляет столбец в таблицу, но не обновляет существующие строки.

Изменение типа столбца

- С ALTER COLUMNFAMILY, вы можете изменить тип столбца любое время после того, как он определен или добавлен в таблицу. Например, если мы решили coupon_code столбец должен хранить коды купонов в виде целых чисел, мы могли бы изменить тип столбца следующим образом:

  ALTER TABLE users ALTER coupon_code TYPE int;

Указание истечения времени

- Обе команды INSERT и UPDATE поддерживает установку в столбец истечения времени (TTL). В приведенном выше примере к jsmith, мы устанавливаем, что пароль столбца истекает через 86400 секунд, или один день.

- Если бы мы хотели продлить срок действия на пять дней, мы могли бы использовать команду UPDATE как показано на рисунке:

  UPDATE users USING TTL 432000 SET 'password' = 'ch@ngem3a' WHERE KEY = 'jsmith';

Удаление столбца метаданных

- Если ваша цель заключается в том, чтобы удалить столбец метаданных полностью, включая имя столбца и тип, вы можете использовать команду *ALTER TABLE <columnFamily> DROP <column>.* Следующая команда удаляет имя и валидатор, не затрагивая или удаляя существующих данных:

ALTER TABLE users DROP coupon_code;

- После выполнения этой команды, клиенты по-прежнему могут добавлять новые столбцы с именами coupon_code в таблицу, но они не будут проверяться до тех пор, пока вы не добавите тип снова.

Создание вторичных индексов

- В этом примере мы будем создавать индекс на state_key и birth_year столбцы в стаблице users.

CREATE INDEX state_key ON users (state);

CREATE INDEX birth_year_key ON users (birth_year);

Удаление столбцов и строк

- Удаление столбца:

DELETE session_token FROM users where KEY = 'jsmith';

- Удаление строки полностью:

DELETE FROM users where KEY ='jsmith';

Удаление таблицы и пространства:

- *DROP COLUMNFAMILY users;*
- *DROP KEYSPACE twissandra;*

Задание №1

Выполнили:
студенты группы БПЗ 1101
Лукин И.Ю.
Хлебников В.А.

Задание

- Произвести вставку 1000000 записей в БД Cassandra и посчитать время, затраченное на эту операцию.

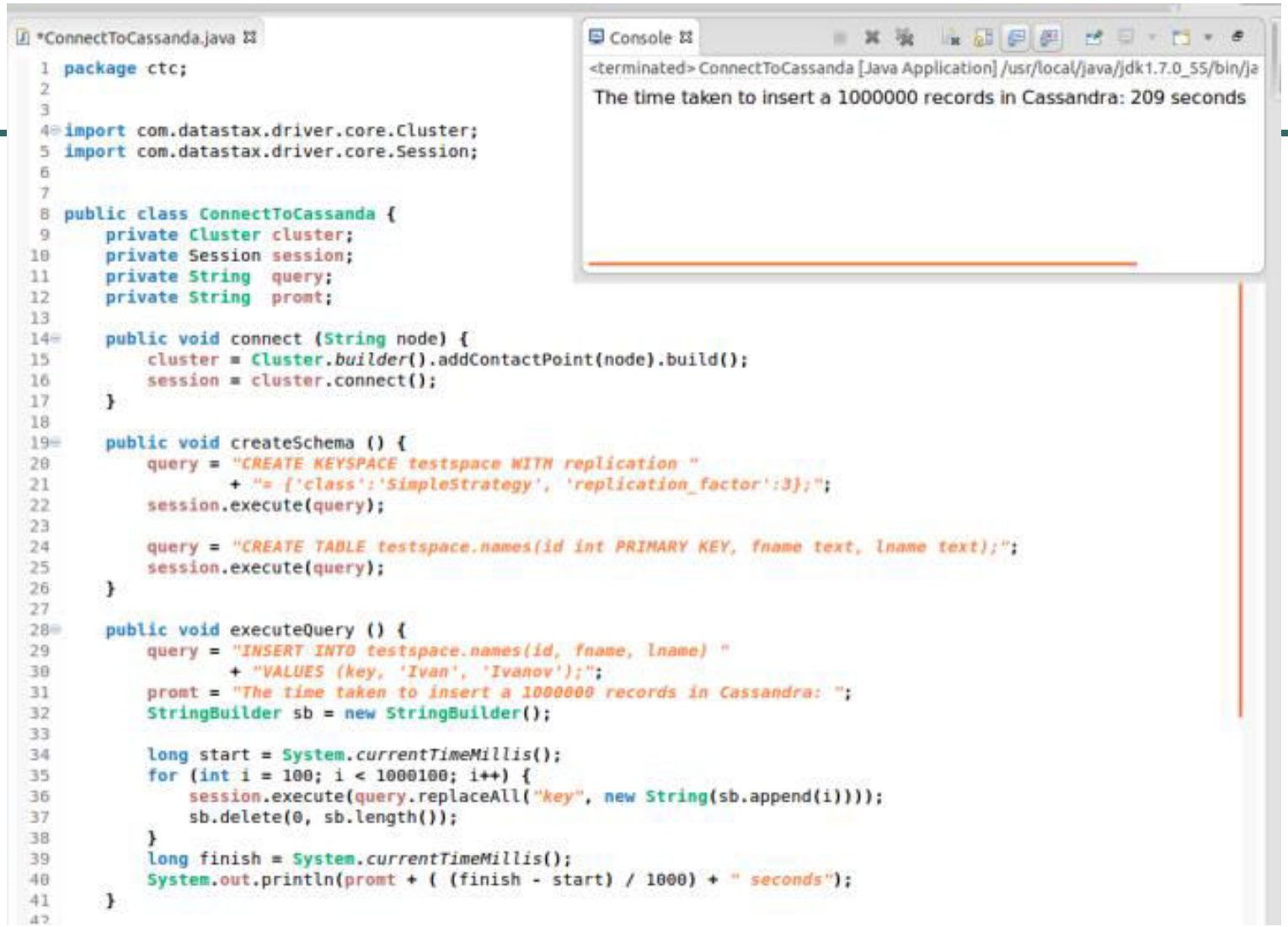
Код программы

```
import com.datastax.driver.core.Cluster;
import com.datastax.driver.core.Session;


public class ConnectToCassanda {
        private Cluster   cluster;
        private Session   session;
        private String    query;
        private String    promt;

        public void connect (String node) {
                cluster = Cluster.builder().addContactPoint(node).build();
                session = cluster.connect();
        }

        public void createSchema () {
                query = "CREATE KEYSPACE testspace WITH replication " +
                        "= {'class':'SimpleStrategy', 'replication_factor':3};";
                session.execute(query);

                query = "CREATE TABLE testspace.names(id int PRIMARY KEY, fname text, lname text);";
                session.execute(query);
        }

        public void executeQuery () {
                query = "INSERT INTO testspace.names(id, fname, lname) "
                                + "VALUES (key, 'Ivan', 'Ivanov');";
                promt = "The time taken to insert a 1000000 records in Cassandra: ";
                StringBuilder sb = new StringBuilder();
```

Продолжение

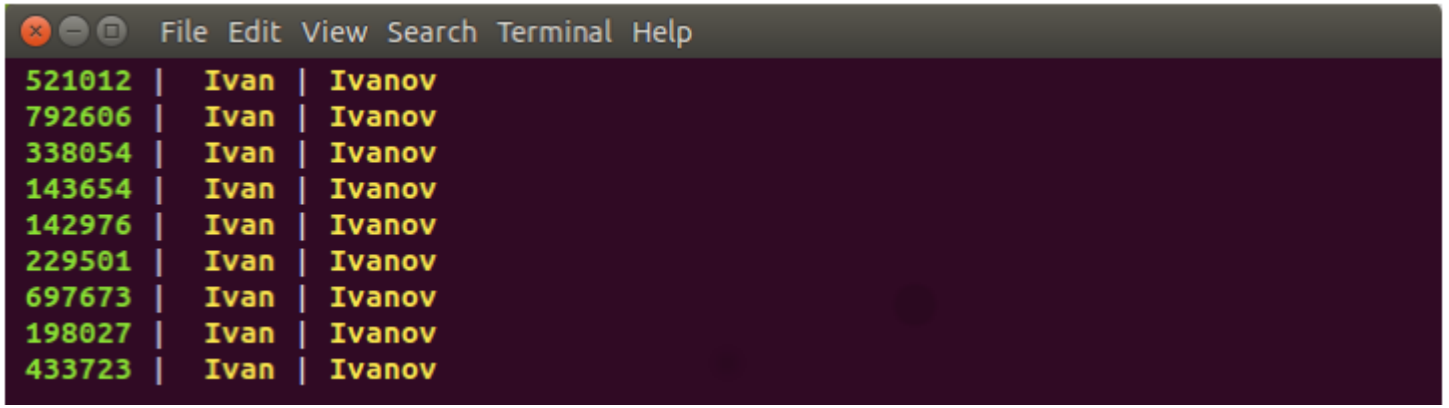
```
            long start = System.currentTimeMillis();
            for (int i = 100; i < 1000100; i++) {
                session.execute(query.replaceAll("key", new String(sb.append(i))));
                sb.delete(0, sb.length());
            }
            long finish = System.currentTimeMillis();
            System.out.println(promt + ( (finish - start) / 1000) + " seconds");
        }

        public void close () {
            cluster.close();
        }

        public static void main (String [] args) {
            ConnectToCassanda client = new ConnectToCassanda();
            client.connect("127.0.0.1");
            client.createSchema();
            client.executeQuery();
            client.close();
        }
}
```

Результат выполнения




```java
package ctc;


import com.datastax.driver.core.Cluster;
import com.datastax.driver.core.Session;


public class ConnectToCassanda {
    private Cluster cluster;
    private Session session;
    private String  query;
    private String  promt;

    public void connect (String node) {
        cluster = Cluster.builder().addContactPoint(node).build();
        session = cluster.connect();
    }

    public void createSchema () {
        query = "CREATE KEYSPACE testspace WITH replication "
                + "= {'class':'SimpleStrategy', 'replication_factor':3};";
        session.execute(query);

        query = "CREATE TABLE testspace.names(id int PRIMARY KEY, fname text, lname text);";
        session.execute(query);
    }

    public void executeQuery () {
        query = "INSERT INTO testspace.names(id, fname, lname) "
                + "VALUES (key, 'Ivan', 'Ivanov');";
        promt = "The time taken to insert a 1000000 records in Cassandra: ";
        StringBuilder sb = new StringBuilder();

        long start = System.currentTimeMillis();
        for (int i = 100; i < 1000100; i++) {
            session.execute(query.replaceAll("key", new String(sb.append(i))));
            sb.delete(0, sb.length());
        }
        long finish = System.currentTimeMillis();
        System.out.println(promt + ( (finish - start) / 1000) + " seconds");
    }
```

- Результат выборки (SELECT * FROM names) :



- Вывод: вставка записей в БД Cassandra происходит быстрее чем в БД PostgreSQL и SQLite, но медленнее чем в БД Berkeley

Задание №2

Выполнили:
студенты группы БПЗ 1101
Кажемский М.А.
Шарыпин Е.М.

Задание

- Использование Python с Apache Cassandra

Использование Python с Apache Cassandra

- Для использования Python с Apache Cassandra необходим модуль pycassa, устанавливамый с помощью пакетного менеджера pip

- pip install pycassa

Подключение и получение ColumnFamily

- Подключимся к кластеру:

```
from pycassa.pool import ConnectionPool
pool = ConnectionPool('Keyspace1‘,['localhost:9160'])
```

- Получение ColumnFamily:

```
from pycassa.columnfamily import ColumnFamily
col_fam = ColumnFamily(pool, 'ColumnFamily1')
```

Добавление данных

- Добавить строку:

```
col_fam.insert('row_key', {'col_name': 'col_val'})
```

- Добавить несколько столбцов в одну строку:

```
col_fam.insert('row_key',
    {'col_name':'col_val','col_name2':'col_val2'})
```

- Добавить несколько строк:

```
col_fam.batch_insert({'row1': {'name1': 'val1', 'name2': 'val2'},
    'row2': {'foo': 'bar'}})
```

Получение данных

- Получить строку:

col_fam.get('row_key')

- Получить несколько столбцов из строки:

col_fam.get('row_key', columns=['col_name', 'col_name2'])

- Получить с пятого по седьмой столбец в строке

col_fam.get('row_key', column_start='5', column_finish='7')

- Получить несколько строк

col_fam.multiget(['row1', 'row2'])

Прочее

- Посчитать количество столбцов в строке:

col_fam.get_count('row_key')

- Подключение нескольких серверов:

pool =
pycassa.ConnectionPool(keyspace='Keyspace 1', server_list=['10.0.0.4:9160', '10.0.0.5:9160'])

Задание №3

Выполнили:
студенты группы БПЗ 1101
Денисов И.А.
Пименов С.А.
Олейникова Т.В.

Задание

- Интегрирование Apache Cassandra в Java

Необходимые утилиты

- Для интеграции нам понадобится следующее программное обеспечение

- Apache Cassandra
- Eclipse IDE
- Maven 2 Eclipse plug-in

Ход работы

- Создадим в Eclipse проект Maven и используем следующие данные:

groupId: com.example.cassandra

artifactId: simple-client

- В Project Viewer выберите

Maven > Add Dependency

- Введите datastax в текстовом поле поиска
- Разверните com.datastax.cassandra и выберите нужную версию

Создание класса

- Создайте новый класс

com.example.cassandra.SimpleClient

Код программы

```
package com.example.cassandra;

import com.datastax.driver.core.Cluster;
import com.datastax.driver.core.Host;
import com.datastax.driver.core.Metadata;

public class SimpleClient {
   private Cluster cluster;

   public void connect(String node) {
      cluster = Cluster.builder()
            .addContactPoint(node).build();
      Metadata metadata = cluster.getMetadata();
      System.out.printf("Connected to cluster: %s\n",
            metadata.getClusterName());
      for ( Host host : metadata.getAllHosts() ) {
         System.out.printf("Datatacenter: %s; Host: %s; Rack: %s\n",
               host.getDatacenter(), host.getAddress(), host.getRack());
      }
   }

   public void close() {
      cluster.shutdown();
   }

   public static void main(String[] args) {
      SimpleClient client = new SimpleClient();
      client.connect("127.0.0.1");
      client.close();
   }
}
```

Запуск:

- Вывод в консоль:

```
Connected to cluster: xerxes
Datatacenter: datacenter1; Host: /127.0.0.1; Rack: rack1
Datatacenter: datacenter1; Host: /127.0.0.2; Rack: rack1
Datatacenter: datacenter1; Host: /127.0.0.3; Rack: rack1
```

Задание №4

Выполнили:
студенты группы БПЗ 1101
Костин Д.В.
Пяткин С.Н.
Назаров Н.Л.

Задание

- Настройка кластера Apache Cassandra

Файл cassandra.yaml

- **cluster_name** - имя нашего кластера. Для всех машин, входящих в кластер, это имя должно быть одинаковым.
- **data_file_directories** - список каталогов, где хранятся данные. Этот каталог должен быть предварительно создан и должны быть права на запись в этот каталог.
- **commitlog_directory** - каталог, куда пишется коммит-лог Кассандры. Этот каталог также должен быть предварительно создан и должны быть права на запись в этот каталог.
- **saved_caches_directory** - каталог, куда пишется сохраненный кеш Кассандры.
- **seeds** - список IP-адресов сидов, т.е. узлов, которые распространяют информацию о кластере. Обычно сидами делают от 5% до 30% машин в кластере.
- **listen_address** - адрес нода с Кассандрой. По умолчанию - localhost. Это нормально, если только запущен один узел с кассандрой.

Single-Node Cluster

- Установите следующие свойства в файле cassandra.yaml

cluster_name: 'MyClusterName‘

initial_token: 0

- Следующие свойства уже правильно настроены для Single-Node, но если вы планируете дальнейшее расширение, рекомендуется их изменить на:

seeds: <IP of node>
listen_address: <IP of node>
rpc_address: 0.0.0.0
endpoint_snitch: RackInferringSnitch|PropertyFileSnitch

Запуск

- Введите в режиме root команду:

 # service cassandra start

Пример развертывания на кластере из двух узлов

- Одна машина под управлением Ubuntu (IP-адрес в локальной сети - 192.168.1.101)
- Вторая - под Windows XP (IP-адрес - 192.168.1.100)

Пример заполнения:

- **cluster_name** - Пример: 'JDevNotes Cluster'
- **data_file_directories** - Значение по умолчанию: /**var**/**lib**/**cassandra**/**data**. На Windows пример: **d:\cassandra_data\data**
- **commitlog_directory** - Значение по умолчанию:/**var**/**lib**/**cassandra**/**commitlog**. На Windows пример: **d:\cassandra_data\commitlog**
- **saved_caches_directory** - По умолчанию:/**var**/**lib**/**cassandra**/**saved_caches**. На Windows: **d:\cassandra_data\saved_caches**
- **seeds** - На кластере из двух машин будет всего лишь один сид - это машина с Ubuntu. Запишем ее адрес сюда: 192.168.1.101. Список сидов должен быть одинаков для всех машин в кластере. На двух наших машинах одинаковой значение - 192.168.1.101.
- **listen_address** - Для Ubuntu - 192.168.1.101, для машины с Windows XP - 192.168.1.100. Сюда нельзя записать адрес 0.0.0.0 - чтобы Кассандра слушала все имеющиеся интерфейсы, т.к. для общения узлов кластера используется Gossip.

Запуск процесса

- *./cassandra -f*

```
INFO 00:20:12,886 Binding thrift service to localhost/127.0.0.1:9160
INFO 00:20:12,888 Using TFramedTransport with a max frame size of 15728640 bytes.
INFO 00:20:12,892 Listening for thrift clients...
INFO 00:22:52,045 Node /192.168.1.100 is now part of the cluster
INFO 00:22:52,838 InetAddress /192.168.1.100 is now UP
INFO 00:22:52,838 Started hinted handoff for endpoint /192.168.1.100
INFO 00:22:52,841 Finished hinted handoff of 0 rows to endpoint /192.168.1.100
```

Multi-Node or Multi-Data Center Cluster

- Пример настройки 6 узлового кластера в файле cassandra.yaml:

- **node0**

```
cluster_name: 'MyDemoCluster'
initial_token: 0
seed_provider:
- seeds: "110.82.155.0,110.82.155.3"
listen_address: 110.82.155.0
rpc_address: 0.0.0.0
endpoint_snitch: RackInferringSnitch
```

- **node1**

```
cluster_name: 'MyDemoCluster‘
initial_token:28356863910078205288614550619314
seed_provider:
-seeds: "110.82.155.0,110.82.155.3"
listen_address: 110.82.155.1
rpc_address: 0.0.0.0
endpoint_snitch: RackInferringSnitch
```

- **node2**

```
cluster_name: 'MyDemoCluster'
initial_token:56713727820156410577229101238628035
seed_provider:
 - seeds: "110.82.155.0,110.82.155.3"
listen_address: 110.82.155.2
rpc_address: 0.0.0.0
endpoint_snitch: RackInferringSnitch
```

- **node3**

```
cluster_name: 'MyDemoCluster'
initial_token:85070591730234615865843651857942052
seed_provider:
      - seeds: "110.82.155.0,110.82.155.3"
listen_address: 110.82.155.3
rpc_address: 0.0.0.0
endpoint_snitch: RackInferringSnitch
```

Продолжение

- **node4**

```
cluster_name: 'MyDemoCluster'
initial_token: 113427455640312821154458202477256070485
seed_provider:
      - seeds: "110.82.155.0,110.82.155.3"
listen_address: 110.82.155.4
rpc_address: 0.0.0.0
endpoint_snitch: RackInferringSnitch
```

- **node5**

```
cluster_name: 'MyDemoCluster'
initial_token: 141784319550391026443072753096570088106
seed_provider:
      - seeds: "110.82.155.0,110.82.155.3"
listen_address: 110.82.155.5
rpc_address: 0.0.0.0
endpoint_snitch: RackInferringSnitch
```

Задание №5

Выполнили:
студенты группы БПЗ 1101
Корниенко П.А.
Канаев С.Д.
Матвеев И.В.

Задание

- Сравнение Apache Cassandra и PostgreSQL

Вставка элементов

- В базу данных вставлялось 1000 элементов и замерялась средняя скорость вставки

БД	Количество записей в БД			
	0	1000	10000	100000
Cassandra	0,000752	0,000315	0,00087	0,000864
Postgre	0,000237	0,000225	0,000184	0,000162

Удаление элементов

- Среднее время удаления одной записи из БД

БД	Количество записей в БД			
	0	1000	10000	100000
Cassandra	0,0005	0,000616	0,000638	0,000878
Postgre	0,000395	0,000706	0,00122	0,009146

Общий анализ

БД	Cassandra	PostgreSQL
Первый выпуск	2008	1989
Язык реализации	Java	C
Сервер ОС	BSD, Linux, OS X, Windows	HP_UX, Linux, OS X, Solaris, Unix, WIndows
Модель БД	Wide column store	Relational DBMS
Поддерживаемые языки программирования	C#, C++, Clojure, Erlang, Go, Haskell, Java, JavaScript, Perl, PHP, Python, Ruby, Scala	.Net, C, C++, Java, Perl, Python, Tcl
Триггеры	Да	Да
Внешние ключи	Нет	Да
Язык запросов	CQL	PSQL
API – интерфейсы	Собственный протокол	С библиотеками, ADO.NEXT, JDBC ODBC
Серверные функции	Нет	Пользовательские функции
Методы репликации	Выбираемый коэффициент репликации	Подчиненная репликация

Задание №6

Выполнили:
студенты группы БПЗ 1101
Кузнецов Е.О.
Зейналов А.С.
Цухникиди П.Ю.

Задание

- Основные команды языка CQL

Особенности

- На первый взгляд напоминает SQL, но существенно урезан в функциональности
- Можно выполнять только простейшие запросы SELECT с выборкой по определённому условию
- Добавление и обновление осуществляется через единое выражение UPDATE, операция INSERT отсутствует
- Из возможностей можно отметить поддержку пространств имён и семейств столбцов, создание индексов через выражение «CREATE INDEX».

Основные команды

- Создание базы данных
- Синтаксис:

<create-keyspace-stmt> ::= CREATE KEYSPACE (IF NOT EXISTS)? <identifier> WITH <properties>

- Пример:

```
CREATE KEYSPACE Excelsior WITH replication = {'class':
    'SimpleStrategy', 'replication_factor' : 3};
```

Удаление базы данных

- Синтаксис

<drop-keyspace-stmt> ::= DROP KEYSPACE (IF EXISTS)? <identifier>

- Пример:

DROP KEYSPACE myApp;

Создание таблицы

- Пример:

```
CREATE TABLE monkeySpecies (
species text PRIMARY KEY,
common_name text,
population varint,
average_size int )
WITH comment='Important biological records'
AND read_repair_chance = 1.0;
```

Удаление таблицы

- Синтаксис:

<drop-table-stmt> ::= DROP TABLE (IF EXISTS)? <tablename>

- Пример:

DROP TABLE worldSeriesAttendees;

Вставка записей

- Синтаксис

<insertStatement> ::= INSERT INTO <tablename>
'(' <identifier> (',' <identifier>) ')'*
VALUES '(' <term-or-literal> (',' <term-or-literal>) ')'*
(IF NOT EXISTS)?
(USING <option> (AND <option>))?*
<term-or-literal> ::= <term>
| <collection-literal>
<option> ::= TIMESTAMP <integer>
| TTL <integer>

- Пример:

INSERT INTO NerdMovies (movie, director, main_actor, year)
VALUES ('Serenity', 'Joss Whedon', 'Nathan Fillion', 2005)
USING TTL 86400;

Создание триггеров

```
CREATE TRIGGER trigger_name ON table_name
USING ‘java_class’
```

Удаление триггеров

```
DROP TRIGGER trigger_name ON table_name
```

```
<select-stmt> ::= SELECT <select-clause>
                  FROM <tablename>
                  ( WHERE <where-clause> )?
                  ( ORDER BY <order-by> )?
                  ( LIMIT <integer> )?
                  ( ALLOW FILTERING )?

<select-clause> ::= DISTINCT? <selection-list>
                  | COUNT '(' ( '*' | '1' ) ')' (AS <identifier>)?

<selection-list> ::= <selector> (AS <identifier>)? ( ',' <selector> (AS <identifier>)? )*
                   | '*'

<selector> ::= <identifier>
             | WRITETIME '(' <identifier> ')'
             | TTL '(' <identifier> ')'
             | <function> '(' (<selector> (',' <selector>)*)? ')'

<where-clause> ::= <relation> ( AND <relation> )*

<relation> ::= <identifier> <op> <term>
             | '(' <identifier> (',' <identifier>)* ')' <op> <term-tuple>
             | <identifier> IN '(' ( <term> ( ',' <term>)* )? ')'
             | '(' <identifier> (',' <identifier>)* ')' IN '(' ( <term-tuple> ( ',' <term-tuple>)* )? ')'
             | TOKEN '(' <identifier> ( ',' <identifer>)* ')' <op> <term>

<op> ::= '=' | '<' | '>' | '<=' | '>='
<order-by> ::= <ordering> ( ',' <odering> )*
<ordering> ::= <identifer> ( ASC | DESC )?
<term-tuple> ::= '(' <term> (',' <term>)* ')'
<op> ::= '=' | '<' | '>' | '<=' | '>='
<order-by> ::= <ordering> ( ',' <odering> )*
<ordering> ::= <identifer> ( ASC | DESC )?
<term-tuple> ::= '(' <term> (',' <term>)* ')'
```

Пример выбора записей

- *SELECT name, occupation FROM users WHERE userid IN (199, 200, 207);*

- *SELECT name AS user_name, occupation AS user_occupation FROM users;*

- *SELECT time, value*

FROM events

WHERE event_type = 'myEvent'

AND time > '2011-02-03'

AND time <= '2012-01-01'

- *SELECT COUNT(*) FROM users;*
- *SELECT COUNT(*) AS user_count FROM users;*

Создание индексов

- Синтаксис:

<create-index-stmt> ::= CREATE (
CUSTOM)? INDEX (IF NOT EXISTS)?
<identifier>? ON <tablename> '('
<identifier> ')' (USING <string> (WITH
OPTIONS = <map-literal>)?)?

- *Пример:*

```
CREATE INDEX userIndex ON NerdMovies (user);
CREATE INDEX ON Mutants (abilityId);
CREATE CUSTOM INDEX ON users (email) USING 'path.to.the.IndexClass';
CREATE CUSTOM INDEX ON users (email) USING 'path.to.the.IndexClass' WITH
    OPTIONS = {'storage': '/mnt/ssd/indexes/'};
```

Используемые источники

- Apache Cassandra - еще одно NoSQL-хранилище» [Электронный ресурс]: Статья -, Владимир Климонтович, - Режим доступа: http://profyclub.ru/docs/172, свободный.
- Моделирование данных в Cassandra 2.0 на CQL3 [Электронный ресурс]: Статья -, Режим доступа: http://habrahabr.ru/post/203200/, свободный.
- Сайт [Электронный ресурс]: Документация, - Режим доступа: http://wiki.apache.org/cassandra, свободный.
- Как устроена apache cassandra [Электронный ресурс]: Статья -, Режим доступа: http://habrahabr.ru/post/155115/, свободный.