# **Лабораторная работа №2. Создание Hive Data warehouse**

# **из текстовых файлов**

Изучается использование системы Hive для создания и использования в качестве хранилища данных (базы данных без удаления и исправления хранимой информации) и движка для трансляции SQL-запросов в mapreduce-задачи

**Вводная часть**

**1. Типы данных Hive**

В создании таблиц могут участвовать следующие типы данных:

* Типы столбцов
* Литералы
* Нулевые значения
* Сложные типы

## **1.1. Типы столбцов**

Тип столбца используются в качестве типов данных столбцов Hive.

### Целые типы

Данные целого типа могут быть определены с помощью интегральных типов данных, **INT**. Когда диапазон данных превышает диапазон INT, нужно использовать BIG**INT**, и если диапазон данных меньше, чем INT, нужно использовать SMALL**INT**. TINYINT меньше SMALLINT.

В таблице представлены различные типы INT данные:

|  |  |  |
| --- | --- | --- |
| Тип | Постфикс | Пример |
| TINYINT | Y | 10Y |
| SMALLINT | S | 10S |
| INT | - | 10 |
| BIGINT | L | 10L |

### Типы строк

Строковые типы можно задать с помощью одиночных (' ') или двойных кавычек (" "). В него входят три формата: Strings, VAR**CHAR** и **CHAR**. В следующей таблице представлены различные строковые типы данных:

|  |  |
| --- | --- |
| Тип данных | Длина |
| STRING | Произвольная |
| VARCHAR | От 1 до 65355 |
| CHAR | 255 |

### Типы времени и дат

TIMESTAMP:

Этот тип поддерживает традиционную UNIX метку времени с дополнительной точностью до наносекунды в формате java.sql.Timestamp "YYYY-MM-DD HH: MM: SS.fffffffff" и "формат гггг-мм-дд чч: мм: ss.ffffffffff".

DATE:

Значения ДАТА описаны в год / месяц / день формат в виде {{YYYY-MM-ДД}}.

INTERVAL

Интервал времени в различных форматах.

### Десятичные\_знаки

Типа DECIMAL в Hive такой же, как и формат Big Десятичный в Java. Он используется для представления произвольной точности. Синтаксис и пример выглядит следующим образом:

DECIMAL(precision, scale)

decimal(10,0)

### Типы Union

UNION представляет собой совокупность разнородных типов данных. Можно создать экземпляр с помощью объединения нескольких разных типов**.** Синтаксис и пример использования приведены ниже:

UNIONTYPE<int, double, array<string>, struct<a:int,b:string>>

{0:1}

{1:2.0}

{2:["three","four"]}

{3:{"a":5,"b":"five"}}

{2:["six","seven"]}

{3:{"a":8,"b":"eight"}}

{0:9}

{1:10.0}

## **1.2. Литералы**

Следующие литералы используются в Hive:

## **С плавающей точкой**

Этот тип не описывает ничего, кроме чисел с десятичной точкой.

## **Десятичного типа**

Этот тип данных отличается только большим диапазоном значений, приблизительно -10 -308 до 10 308.

## **1.3. Нулевое значение**

Отсутствующие значения представляются специальным значением NULL.

## **1.4. Сложные типы**

Следующие типы сложных данных имеются в Hive:

### Массивы

Массивы в Hive используются так же, как они используются в Java.

Syntax: ARRAY<data\_type>

### Карты

Карты в Hive похожи на Java Maps.

Syntax: MAP<primitive\_type, data\_type>

### Структуры

В Hive структуры используются по аналогии со сложными данными с комментариями.

Syntax: STRUCT<col\_name : data\_type [COMMENT col\_comment], ...>

Подробнее о типах данных в Hive можно прочитать в энциклопедии HIVE по адресу, <https://cwiki.apache.org/confluence/display/Hive/Home> , поиск по сайту «LanguageManual Types»

**2**. **Основные операции Hive**

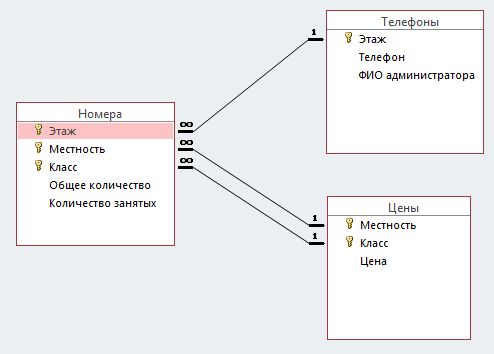
Операции над данными в Hive на языке HQL схожи с аналогичными операциями в SQL системах:

CREATE EXTERNAL TABLE создать внешнюю таблицу

SELECT… FROM …WHERE … выбрать (что?) из (таблицы) где (условие)

JOIN соединить две или более таблиц

# **Схема (структура) базы данных, с которой производится работа**



Для выполнения дальнейшей работы необходимо подготовить три текстовых файла, в которых будут храниться данные. По одному файла на таблицу в формате:

<значение поля>|< значение поля >|< значение поля >|< значение поля >, например, для таблицы Цены файл с данными может выглядеть так:

3|1|3500

3|2|2500

2|1|3000

**Выполнение лабораторной работы**

**3. Создание базы данных**

3.1. Установка HDFS и Hive.

Для начала работы с Hive (рис. 1) необходимо нажать на закладку Hue (также, как мы делали на предыдущей лабораторной работе), логин/пароль cloudera/cloudera

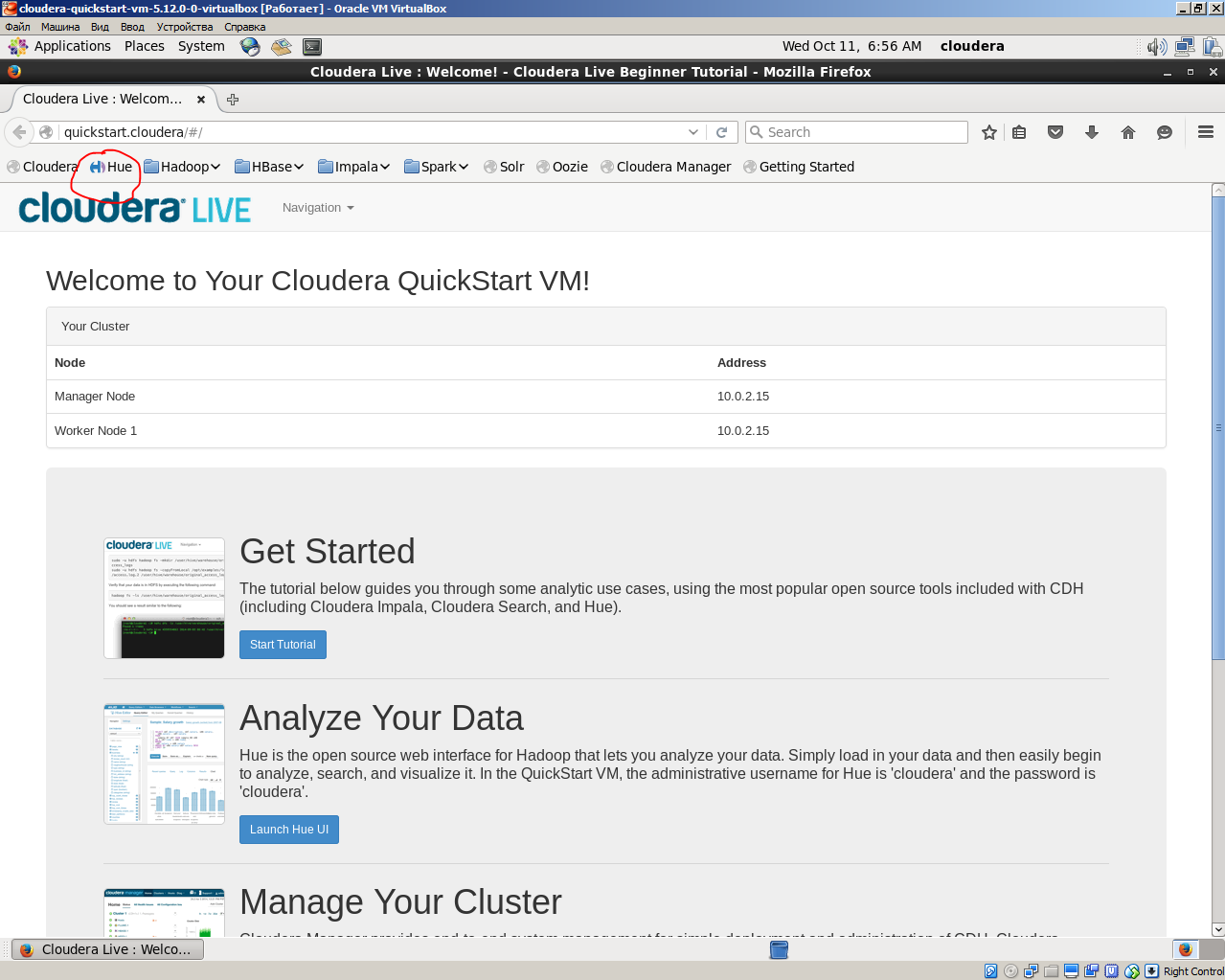


Рис.1.

Далее необходимо выбрать (рис. 1) Query Editor -> Hive и Вы у цели.

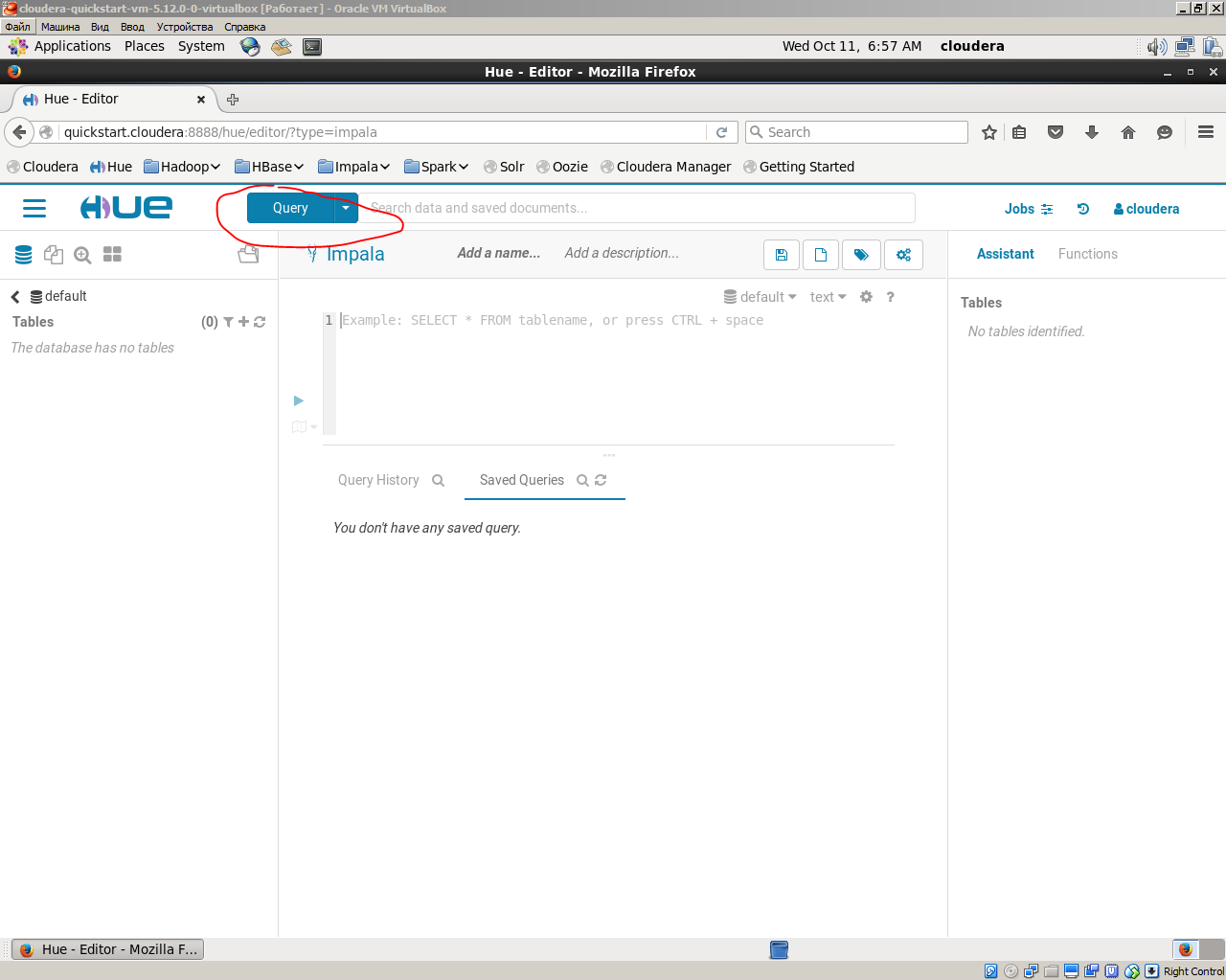


Рис.2.

На рабочем столе (директория /home/cloudera/Desktop/) нужно создать три файла price, room и phone\_number и заполнить их содержимым из одноименных файлов репозитория kolychev1995/HiveLab, расположеных в директории data.

3.2. Создание директории в HDFS и копирование из локальной файловой системы файла с данными в директорию HDFS (команды выполняются в терминале).

hadoop fs -mkdir /user/hive/warehouse/price/

hadoop fs -copyFromLocal /home/cloudera/Desktop/price /user/hive/warehouse/**price**/

3.3. Создание таблиц

3.3.1. Создание таблицы **PRICE** «над» файлом, скопированным в HDFS.

DROP TABLE IF EXISTS price;

CREATE EXTERNAL TABLE PRICE (

area tinyint,

class tinyint,

price smallint

)

ROW FORMAT DELIMITED FIELDS TERMINATED BY '|'

LOCATION '/user/hive/warehouse/price/';

3.3.2. Создание таблицы **ROOM** «над» файлом, скопированным в HDFS.

hadoop fs -mkdir /user/hive/warehouse/room/

hadoop fs -copyFromLocal /home/cloudera/Desktop/room /user/hive/warehouse/**room**/

drop TABLE IF EXISTS room;

CREATE EXTERNAL TABLE ROOM (

level tinyint,

area tinyint,

class tinyint,

quantity smallint,

quantity\_taken\_room smallint

)

ROW FORMAT DELIMITED FIELDS TERMINATED BY '|'

LOCATION '/user/hive/warehouse/**room**/';

3.3.3. Создание таблицы **PHONE\_NUMBER** «над» файлом, скопированным в HDFS

hadoop fs -mkdir /user/hive/warehouse/phone\_number/

hadoop fs -copyFromLocal /home/cloudera/Desktop/phone\_number /user/hive/warehouse/**phone\_number**/

drop TABLE IF EXISTS PHONE\_NUMBER;

CREATE EXTERNAL TABLE PHONE\_NUMBER (

level tinyint,

phone\_number int,

admin\_name string

)

ROW FORMAT DELIMITED FIELDS TERMINATED BY '|'

LOCATION '/user/hive/warehouse/**phone\_number**/';

База данных создана, информация в нее загружена. Обратите внимание, что в отличии от создания базы данных в традиционных реляционных базах данных, в Hive не указываются ни первичные ключи таблиц, ни связи между сущностями, ни ограничения.

# **4. Формирование запросов к базе данных**

## **4.1. Простые запросы на поиск**

4.1.1. Вывести список цен проживания в двухместных номерах разного класса:

SELECT \*

FROM price

WHERE area=2;

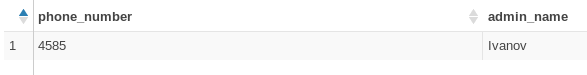


4.1.2. Вывести телефон и фамилию администратора первого этажа

SELECT phone\_number, admin\_name

FROM phone\_number

WHERE level=1;



4.1.3. Вывести количество одноместных номеров первого класса по этажам.

SELECT level, quantity

FROM room

WHERE area = 1 and class = 1;



## **4.2. Более сложные запросы с операцией Join**

4.2.1. Вывести фамилии администраторов и номера телефонов этажей, на которых есть незанятые одноместные номера:

SELECT DISTINCT phone\_number.admin\_name, phone\_number.phone\_number

FROM phone\_number INNER JOIN room ON phone\_number.level = room.level

WHERE room.area = 1 And room.quantity > room.quantity\_taken\_room;

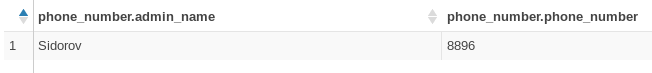


4.2.2. Вывести фамилии администраторов и номера телефонов этажей, на которых есть больше девяти свободных двуместных номеров:

SELECT DISTINCT phone\_number.admin\_name, phone\_number.phone\_number

FROM phone\_number INNER JOIN room ON phone\_number.level = room.level

WHERE room.area = 2 And (room.quantity - room.quantity\_taken\_room) > 9;



4.2.3. Вывести фамилию администратора и номера телефонов этажа, на котором больше всего трехместных номеров

SELECT DISTINCT phone\_number.admin\_name, phone\_number.phone\_number

FROM phone\_number INNER JOIN room ON phone\_number.level = room.level

WHERE room.area = 3 And room.quantity = (

select max(quantity)

from room

where area = 3

);

Этот запрос может не заработать в текущей версии Hive, хотя здесь используется совсем несложный подзапрос.

## **4.3. Запросы на поиск с условием в одной таблице**

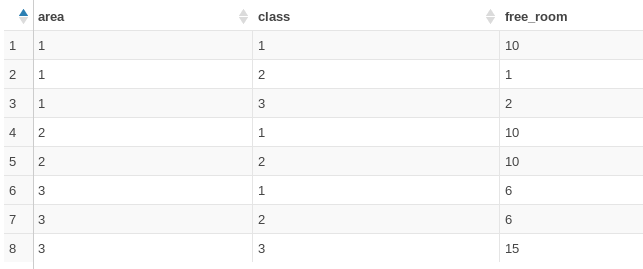
4.3.1. Сформировать количественное распределение незанятых номеров по

местности и классу:

SELECT area, class, Sum(quantity-quantity\_taken\_room) AS free\_room

FROM room

GROUP BY area, class;



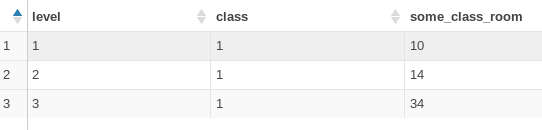
4.3.2. Сформировать количественное распределение номеров заданного класса по этажам:

SELECT level, class, SUM(quantity) AS some\_class\_room

FROM room

WHERE class=1

GROUP BY level, class;



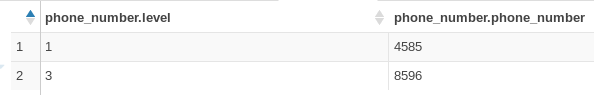
## **4.4. Более сложный поиск с операцией Join (выполнять по желанию)**

4.4.1. Вывести телефоны этажей, на которых расположены одноместные номера люкс:

SELECT DISTINCT phone\_number.level, phone\_number.phone\_number

FROM phone\_number INNER JOIN room ON phone\_number.level = room.level

WHERE area=1 AND class=1;

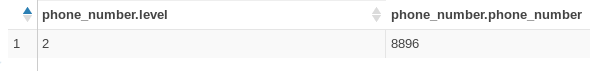


4.4.2. Вывести телефоны этажей, на которых расположены двуместные номера второго класса:

SELECT DISTINCT phone\_number.level, phone\_number.phone\_number

FROM phone\_number INNER JOIN room ON phone\_number.level = room.level

WHERE area=2 AND class=2



4.4.3. Вывести телефоны этажей, на которых расположены трехместные номера второго класса или третьего класса:

SELECT DISTINCT phone\_number.level, phone\_number.phone\_number

FROM phone\_number INNER JOIN room ON phone\_number.level = room.level

WHERE area=3 AND (class=2 OR class=3);

