# **Лабораторная работа №2. *Использование Apache Hive для создания Data WareHouse из текстовых файлов***

**Цель работы:** изучение возможностей системы *Apache Hive* для создания и использования хранилища данных (базы данных с ограниченной поддержкой операции удаления и исправления хранимой информации), а также для трансляции *SQL*-запросов в *mapreduce*-задачи.

**2.1. Типы данных и основные операции Apache Hive**

*Apache Hive* - это система управления базами данных на основе платформы *Hadoop*. Она позволяет выполнять запросы, агрегировать и анализировать данные, хранящиеся в *Hadoop*.

*Apache Hive* был создан корпорацией *Facebook,* и передан под открытой лицензией в собственность фонду *Apache Software Foundation*. *Hive* обладает возможностями:

* работы с данными используя *SQL*-подобный язык запросов;
* поддержки различных форматов хранения данных;
* работы напрямую с файловой системой *HDFS* и *Apache HBase*;

В создании таблиц *Hive* могут участвовать следующие типы данных: столбцы; литералы; нулевые значения; сложные типы.

## ***Столбцы.*** Тип столбца используется в качестве типа данных.

***Целые.***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 2.1 представлены различные типы INT данных.

***Таблица 2.1 Целые типы.***

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

### *Строки*

Строковые типы можно задать с помощью одиночных (' ') или двойных кавычек (" "). В него входят три формата: *Strings*, *VAR****CHAR*** и ***CHAR***. В таблице 2.2 представлены различные строко-

вые типы данных.

***Таблица 2.2 Строковые типы.***

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

### *Время и даты (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*

Тип *DECIMAL* в *Hive* такой же, как и формат *Big* Десятичный в *Java*. Он используется для представления произвольной точности. Его синтаксис и пример выглядят так:

DECIMAL(precision, scale)

decimal(10,0)

### *Union*

Тип *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}

## ***Литералы.*** В Hive используются следующие литералы: c плавающей точкой; десятичного типа (от -10-308 до 10308);

## ***Нулевое значение.*** Отсутствующие значения представляются специальным значением NULL.

***Сложные.*** 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*»

***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.

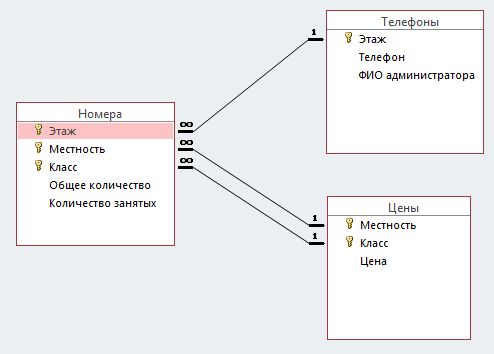
**2.2. Подготовка текстовых файлов и загрузка их в *Apache Hive***

В соответствии со схемой базы данных (рис. 2.1), для выполнения дальнейшей работы необходимо подготовить три текстовых файла, по одному на каждую таблицу. В этих файлах будут храниться данные в формате: <значение поля>|< значение поля >|< значение поля >|< значение поля >. Текстовые файлы удобно создать в «простом» текстовом редакторе (например, *notepad*++), чтобы при сохранении текстовый файл не содержал дополнительной информации. Также лучше подготавливать файлы с данными непосредственно в виртуальной машине, так как в ОС семейств Windows и Linux отличаются символы переноса строк. Для таблицы Цены (*Price*) файл с данными может выглядеть так:

3|1|3500

3|2|2500

2|1|3000

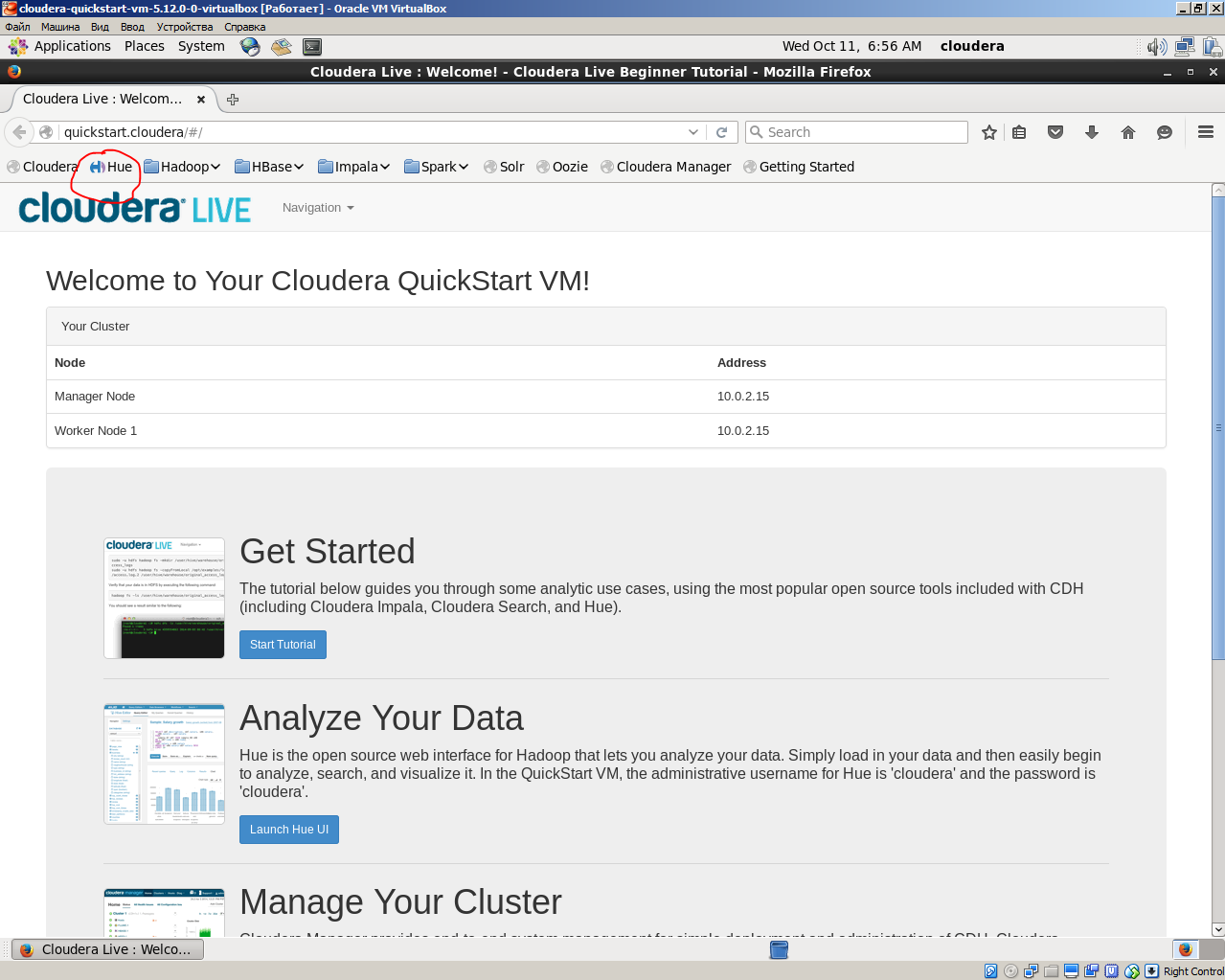


# ***Рис. 2.1.******Схема (структура) базы данных для работы в Hive.***

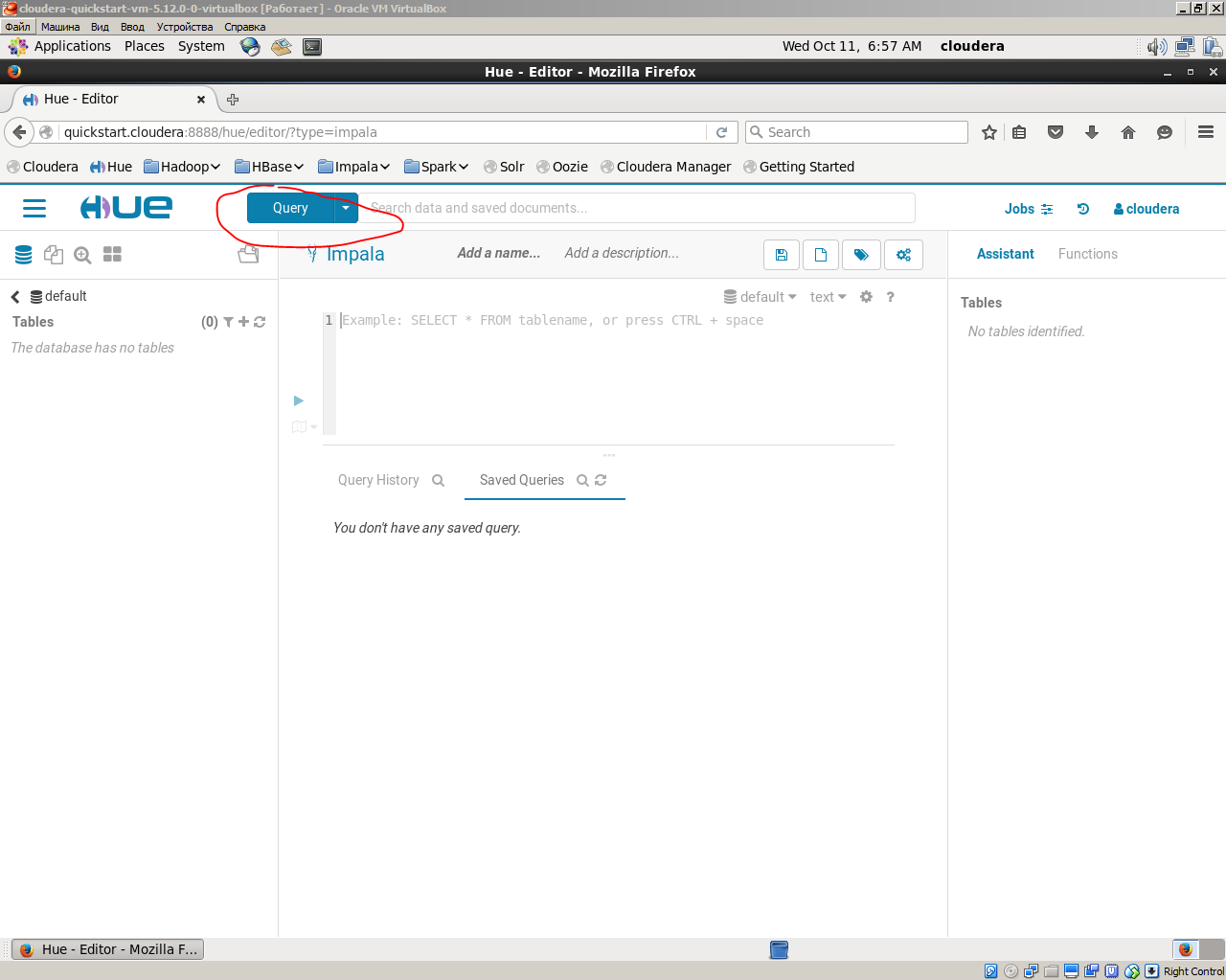
**Установка** ***HDFS*** **и** ***Hive***

Для начала работы с *Hive* (рис. 2.2) необходимо нажать на закладку *Hue* (так же, как мы делали на предыдущей лабораторной работе), логин/пароль здесь ***cloudera/cloudera.*** Далее выбираем (см. рис. 2.3) Query Editor -> Hive.

Закачка текстовых файлов описаний таблиц ***PRICE, ROOM*** и ***PHONE\_NUMBER*** должна быть выполнена заранее.



***Рис.2.2. Выбор закладки Hue.***



***Рис.2.3. Выбор Query Editor*.**

Отметим, что система Hadoop имеет собственную файловую систему HDFS, которая создана специально для работы с большими данными, поэтому для начала работы с данными их нужно скопировать из локальной файловой системы, где развернут Hadoop, в HDFS. Команды создания директории в *HDFS* и копирование из локальной файловой системы файлов с данными в директорию *HDFS* выполняются в терминале.

**Создание таблиц**

Создаем директорию в HDFS, где будут расположены данные для таблицы ***PRICE*** (команды выполняются в терминале):

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

Копируем данные из локальной файловой системы в HDFS (первый путь в локальной ФС, второй в HDFS):

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

Создадим таблицу ***PRICE*** «над» каталогом, в который скопированы данные из локальной файловой системы (выполняется в редакторе запросов, рис.2.3):

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/';

Далее по аналогии создадим таблицу ***ROOM*** «над» каталогом, в который скопированы данные из локальной файловой системы

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**/';

И, наконец, создадим таблицу ***PHONE\_NUMBER*** «над» каталогом, в который скопированы данные из локальной файловой системы

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

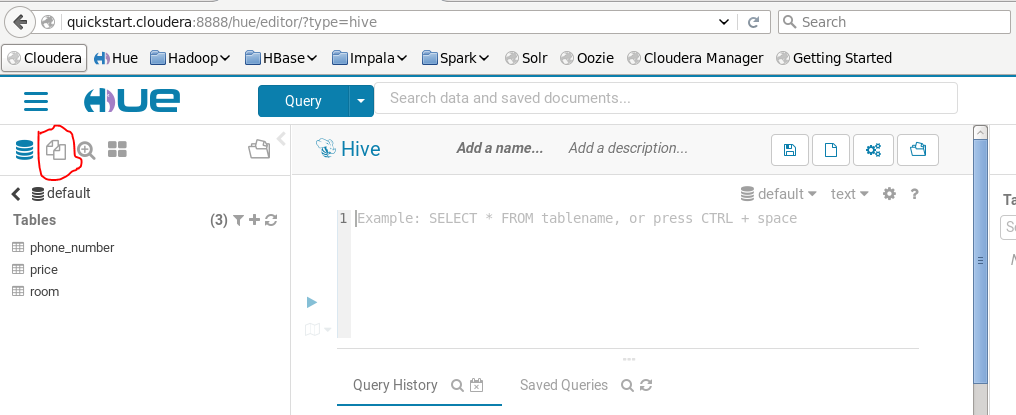
)

ROW FORMAT DELIMITED FIELDS TERMINATED BY '|'

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

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

Просмотреть каталоги и файлы в HDFS можно в меню редактора запросов, для этого необходимо нажать на вкладку HDFS слева от окна редактирования запросов (рис.2.4). Там доступны обычные действия с файлами и каталогами, просмотр, изменение, удаление и т.п.



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

## ***Простые запросы на поиск***

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

SELECT \*

FROM price

WHERE area=2;

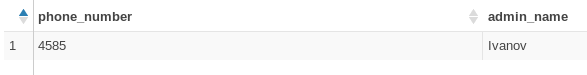


2. Вывести телефон и фамилию администратора первого этажа

SELECT phone\_number, admin\_name

FROM phone\_number

WHERE level=1;



3. Вывести количество одноместных номеров первого класса по этажам.

SELECT level, quantity

FROM room

WHERE area = 1 and class = 1;



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

4. 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;

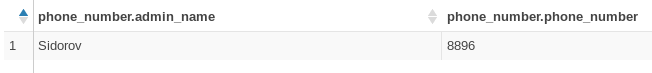


5. Вывести фамилии администраторов и номера телефонов этажей, на которых есть больше девяти свободных двуместных номеров:

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;



6. 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.

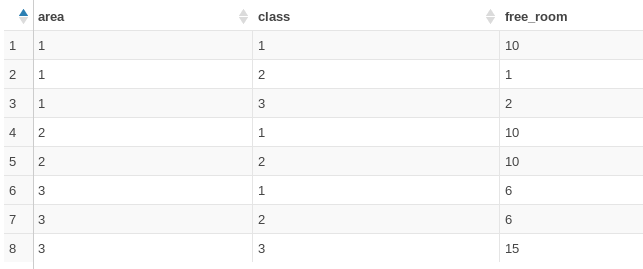
## ***Запросы на поиск с условием в одной таблице***

7. 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;



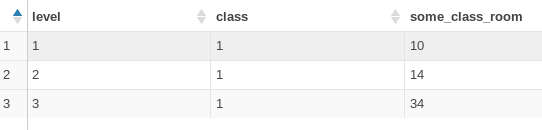
8. Сформировать количественное распределение номеров заданного класса по этажам:

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

FROM room

WHERE class=1

GROUP BY level, class;



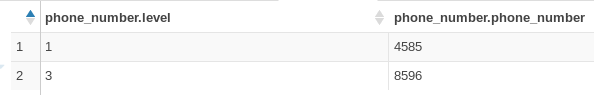
## ***Более сложный поиск с операцией Join (выполнять по желанию)***

9. 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;

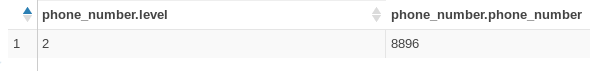


10. Вывести телефоны этажей, на которых расположены двуместные номера второго класса:

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

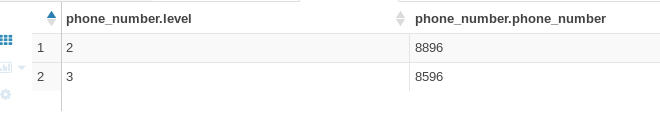


11. Вывести телефоны этажей, на которых расположены трехместные номера второго класса или третьего класса:

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);



**2.4. Контрольные вопросы**

1. Во что Hive преобразует запросы?
2. Перечислите целые типы данных Hive. В чем их отличия?
3. Перечислите строковый типы данных Hive. В чем их отличия?
4. Какие типы данных используются для задания времени и дат?
5. Каким значением представляются отсутствующие значения?
6. Перечислите сложные типы данных Hive. В чем их отличия?
7. В чем особенность типа данных UNION?
8. Как называется язык запросов, используемый в Hive?
9. Каковы основные операции над данными доступные в Hive?
10. Какие шаги выполняются для создания таблицы в Hive?
11. В чём состоят основные отличия при создании БД в Hive и традиционных СУБД?
12. Какие типы запросов не поддерживает Hive?
13. Каким образом можно соединить две таблицы в Hive?