Содержание

[Описание предметной области 3](#_Toc530213735)

[Задание 3](#_Toc530213736)

[База данных 3](#_Toc530213737)

[Процесс выполнения 3](#_Toc530213738)

[Приложение 5](#_Toc530213739)

[Main.py 5](#_Toc530213740)

[auto.txt 6](#_Toc530213741)

[Selling.txt 6](#_Toc530213742)

[Result.txt 6](#_Toc530213743)

Описание предметной области

Apache Hive — система управления базами данных на основе платформы Hadoop. 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. На сегодняшний день эта система используется компанией Netflix и доступна в Amazon Web Services через Amazon Elastic MapReduce.

Hive представляет из себя движок, который превращает SQL-запросы в цепочки map-reduce задач. Движок включает в себя такие компоненты, как Parser(разбирает входящие SQL-запрсоы), Optimimer(оптимизирует запрос для достижения большей эффективности), Planner (планирует задачи на выполнение) Executor(запускает задачи на фреймворке MapReduce.

Для работы hive также необходимо хранилище метаданных. Дело в том что SQL предполагает работу с такими объектами как база данных, таблица, колонки, строчки, ячейки и тд. Поскольку сами данные, которые использует hive хранятся просто в виде файлов на hdfs — необходимо где-то хранить соответствие между объектами hive и реальными файлами.

В качестве metastorage используется обычная реляционная СУБД, такая как MySQL, PostgreSQL или Oracle.

Задание

Разработать схему базы данных, создать базу данных в Hive. Применить partitioning, bucketing. Разработать 5 запросов с JOIN. 1 из запросов реализовать на Map-Reduce на любом языке программирования.

База данных

База данных содержит 2 таблицы. Первая таблица содержит название марок автомобилей, вторая количество проданных автомобилей.

id name id sum

-- ---- -- ----

1 bmw 1 250

2 opel 5 450

3 kia 3 300

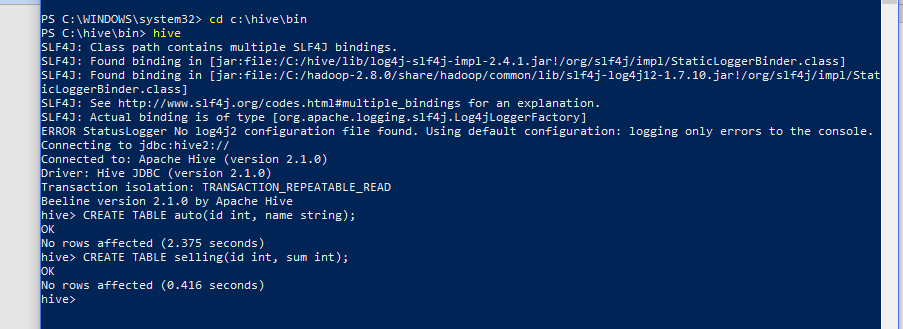
4 audi 6 400

Процесс выполнения

Запуск сервера derby

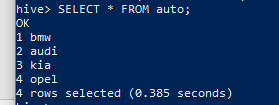


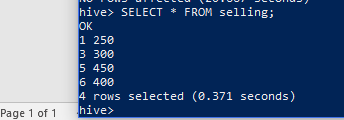
Запустим Hive и создадим таблицы



Добавить данные в таблицы с помощью команды INSERT INTO table(id, name) values(1,"usr");

Вывод информации из таблиц

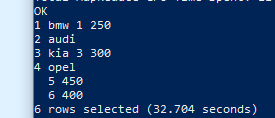




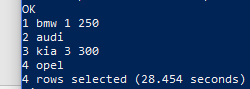
1) SELECT \* FROM auto INNER JOIN selling ON auto.id = selling.id;



2) SELECT \* FROM auto FULL JOIN selling ON auto.id = selling.id;



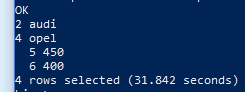
3) SELECT \* FROM auto LEFT OUTER JOIN selling ON auto.id = selling.id;



4) SELECT \* FROM auto LEFT OUTER JOIN selling ON auto.id = selling.id WHERE selling.id IS null;



5) SELECT \* FROM auto FULL OUTER JOIN selling ON auto.id = selling.id WHERE auto.id IS null OR selling.id IS null;



Пример результата работы запроса INNER JOIN на основе MapReduce и языка программирования python:

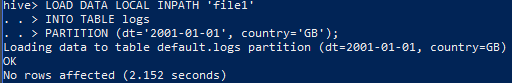


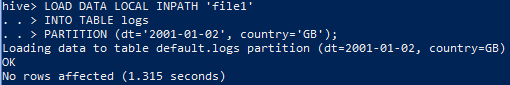
Partition

Создание таблицы с partition

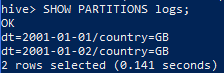


Заполнение таблицы из файла





Partitions таблицы logs

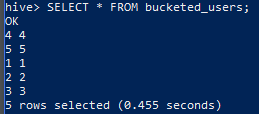


Bucket

Создание таблицы с разделением на 4 buckets



Все данные в таблице



Вывод данных через buckets

















Приложение

## mapper.py

#!/usr/bin/python

import sys

if \_\_name\_\_ == '\_\_main\_\_':

for line in sys.stdin:

if line:

print('\t'.join(x for x in line.strip().split(" ") if x != ''))

## reducer.py

#!/usr/bin/python

import sys

def reduce(\*args):

result = ''

\_map\_result\_first = args[0]

\_map\_result\_second = args[1]

for key1 in \_map\_result\_first.keys():

if key1 in \_map\_result\_second:

result += str(\_map\_result\_first[key1]) + ' ' + str(key1) + ' ' + str(\_map\_result\_second[key1]) + '\n'

print(result)

if \_\_name\_\_ == "\_\_main\_\_":

STR = '-\t-\n'

map\_result\_first = {}

map\_result\_second = {}

for line in sys.stdin:

if line == STR:

continue

temp = line.replace('\n', '').split('\t')

if temp[1].isalpha():

map\_result\_first.update({temp[0]: temp[1]})

continue

map\_result\_second.update({temp[0]: temp[1]})

reduce(map\_result\_first, map\_result\_second)

## runPy.sh

#!/bin/bash

hdfs dfs -rm -r /output

yarn jar c:\hadoop-2.8.0\share\hadoop\tools\lib\hadoop-streaming-\*.jar \

-input /input/file1 \

-output /output \

-file .\mapper.py \

-file .\reducer.py \

-mapper "python .\mapper.py" \

-reducer "python .\reducer.py"

hadoop fs -text /output/part-00000

## runPy.ps1

hdfs dfs -rm -r -f /output

yarn jar c:\hadoop-2.8.0\share\hadoop\tools\lib\hadoop-streaming-\*.jar -input /input/file1 -output /output -file .\mapper.py -file .\reducer.py -mapper "python .\mapper.py" -reducer "python .\reducer.py"

hadoop fs -text /output/part-00000

## file1

1 bmw

2 audi

3 kia

4 opel

- -

1 250

3 300

5 450

6 400

- -