Для начала нужно зайти в hive командой ‘hive’

Запрос без входа в hive: hive –e “…”

Столбцы не всегда показываются, в зависимости от настроек, поэтому команда

Посмотреть столбцы:

describe database.tablename;

Что бы создать таблицу:

create table nameCount( name String, count int, timelast timestamp)

\*PARTITIONED BY (date STRING)

\*ROW FORMAT DELIMITED #связаны

\*FIELDS TERMINATED BY ','#связаны

\*LINES TERMINATED BY '\n'#связаны

\*STORED AS TEXTFILE;

\* - необязательные

Вставить из таблицы в другую таблицу:

INSERT INTO TABLE nameCount SELECT name, 1, datelog FROM namelogs;

1 –нет такого столбца поэтому добавиться как 1 int.

Загружайте данные непосредственно из локального в hive:

LOAD DATA LOCAL INPATH 'data/abc.csv' “OVERWRITE” INTO TABLE <table name>;

LOCAL – если убрать то будет из HDFS

Перезаписать таблицу: (предпочтительно в Hive)

INSERT overwrite TABLE nameCount SELECT name, 1, datelog FROM namelogs;

**External таблицы**

После этого таблицей можно пользоваться точно так же как и обычными таблицами hive. Самое удобное в этом, что вы можете просто скопировать файл в нужную папочку в hdfs, а hive будет автоматом подхватывать новые файлы при запросах к соответствующей таблице. Это очень удобно при работе например с логами.

**Партиция (partition)**

Так как hive представляет из себя движок для трансляции SQL-запросов в mapreduce-задачи, то обычно даже простейшие запросы к таблице приводят к полному сканированию данных в этой таблицы. Для того чтобы избежать полного сканирования данных по некоторым из колонок таблицы можно произвести партиционирование этой таблицы. Это означает, что данные относящиеся к разным значениям будут физически храниться в разных папках на HDFS.

PARTITIONED BY (birth\_year int, birth\_month string)

//столбцов birth\_year, birth\_month не будет в таблице

**Static Partitioning**

При заливке данных в такую таблицу необходимо явно указать, в какую партицию мы заливаем данные:

LOAD DATA INPATH '/user/root/sample.txt' OVERWRITE

INTO TABLE employee\_partitioned

PARTITION (birth\_year=1998, birth\_month='May');

Получится следующая дериктория в hdfs:

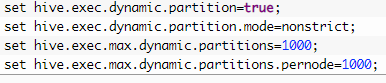
/user/hive/warehouse/employee\_partitioned/birth\_year=1998/birth\_month=May/

**Dynamic Partitioning**

Вместо загрузки каждого раздела с помощью одной инструкции SQL, как показано выше, что приведет к написанию большого количества инструкций SQL для огромного количества разделов, Hive поддерживает динамическое разделение, с помощью которого мы можем добавлять любое количество разделов с помощью одного выполнения SQL. Hive автоматически разбивает наши данные на отдельные файлы разделов на основе значений ключей разделов, присутствующих во входных файлах.

// при добавлении последний столбец идет в партицию, поэтому их имена должны совпадать

// так же должны быть включены флаги hive



INSERT OVERWRITE TABLE demo\_tab PARTITION (land)

SELECT stadt, geograph\_breite, id, t.country,t.id as land

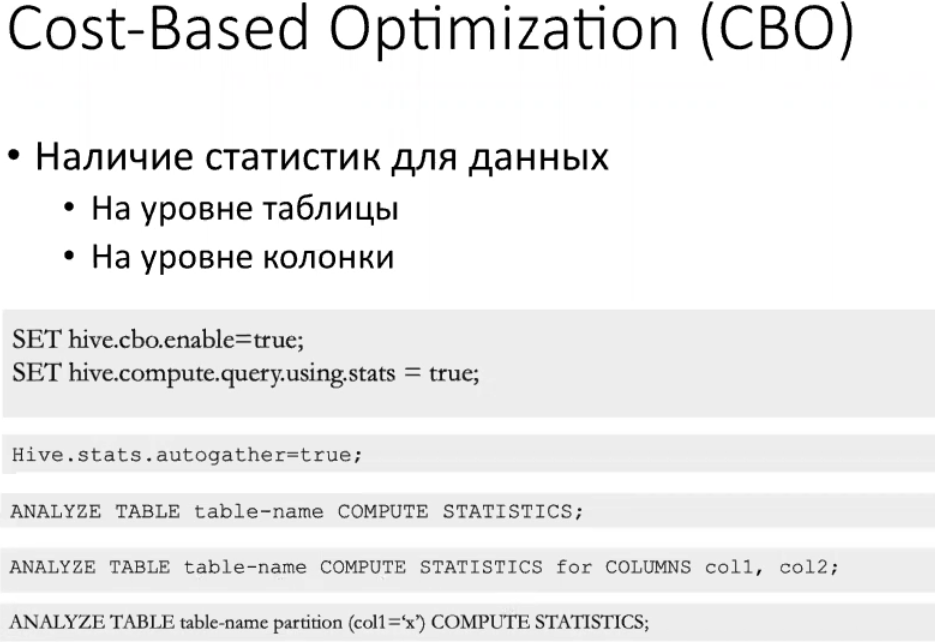
FROM demo\_stg t;

**Бакет**

Партиционирование помогает сократить время обработки, если обычно при запросах известны ограничения на значения какого-либо столбца. Однако оно не всегда применимо. Например — если количество значений в столбце очень велико. Напрмер — это может быть ID пользователя в системе, содержащей несколько миллионов пользователей.

CLUSTERED BY(eid) INTO 10 BUCKETS

В этом случае на помощь нам придет разделение таблицы на бакеты. В один бакет попадают строчки таблицы, для которых значение совпадает значение хэш-функции вычисленное по определенной колонке.



**Формат хранения файлов**

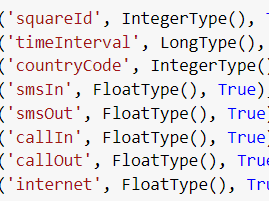
Формат хранения файлов hive включает в себя следующие категории:

**（1）、TEXTFILE  
（2）、SEQUENCEFILE  
（3）、RCFILE  
(4) ORCFILE (appears after 0.11)  
（5）、PARQUET**

Что бы нормально работало нельзя вставлять из локального хранилища (исключение textfile – через него можно вставлять в таблицы другого формата)

**Задание**

Есть milan\_csv со след. Полями



1. Сделать партицирование (static – искусственно по временным интервалам)

(dynamic-по timeinterval)

1. Записать в виде orc
2. Поработать с json

INSERT INTO TABLE milan\_part\_sqr\_orc SELECT squareid, countrycode,smsin,smsout,callin,callout,internet,timeinterval from milan\_csv;

CREATE TABLE milan\_part\_sqr\_orc (squareid int, countrycode int, smsin float, smsout float, callin float, callout float, internet float) PARTITIONED BY (timeinterval bigint) row format delimited fields terminated by '\t' STORED AS orc;

**JSON**

1. Обычной загрузкой каждая строчка будет отдельным объектом
2. Select get\_json\_object(strok, ‘$.ID’) from jspc; - вытащить из каждой строчки поля ID (регистр имеет значение)
3. $: Возвращает корневой объект

.: Вовзращает объект-ребенок

[]: Обращение по индексу в массиве

\*: Wildcard

1. Делать для каждого объекта новую строчку в исходнике

**HINT’S**

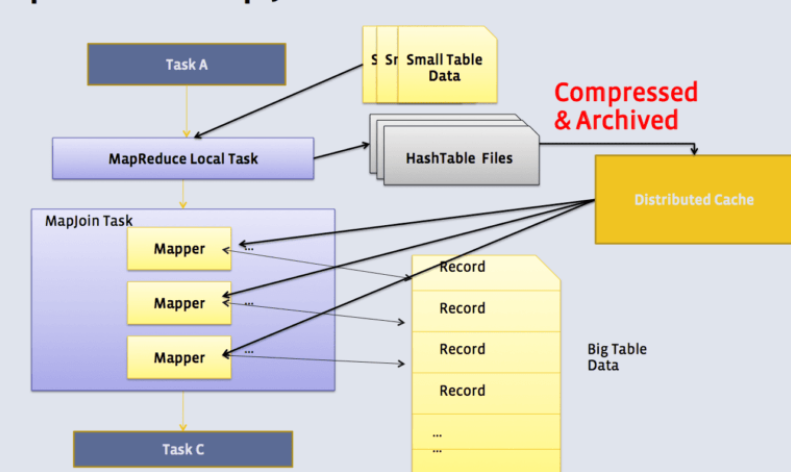
*Позволяют оптимизировать запросы, джоины*

**MapJoin (broadcast join)**

**-**передача таблицы в память каждой ноды для джойнов:

SELECT /\*+ MAPJOIN(**time\_dim**) \*/ COUNT(**\***) from

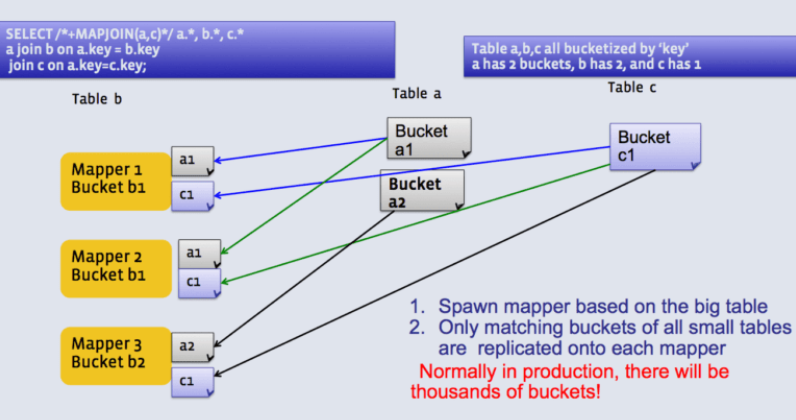
store\_sales JOIN time\_dim on (**ss\_sold\_time\_sk** = t\_time\_sk)



**Bucket Map Join**

– соединение сегментами (бакетами)

Объединение выполняется только в **mapper**. Mapper обработки сегмента 1 для таблицы A будет получать только сегмент 1 из таблицы B.

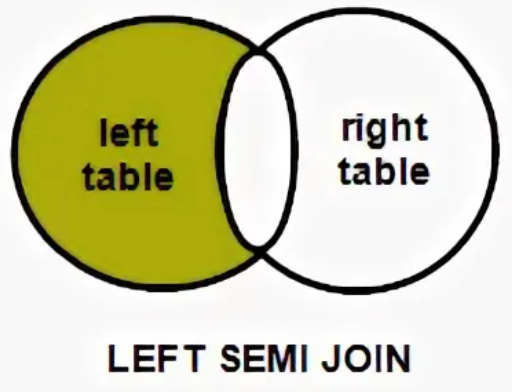


Когда все таблицы большие b количество сегментов в одной таблице должно быть кратно количеству сегментов в другой таблице. Таблицы должны быть сегментированы по тем же полям, которые используются для соединения.

**LEFT SEMI JOIN**

В HIVE не работают конструкции IN, NOT IN, EXISTS

= > поэтому используют LEFT SEMI JOIN:



**Картинка дерьмо**

**select** \*

**from** Customers C

**where** **exists** (

**select** \*

**from** Sales S

**where** S.Cust\_Id = C.Cust\_Id

)

Аналог:

**select** \*

**from** Customers C

**left semi join** Sales\_S **on** S.Cust\_Id = C.Cust\_Id

SELECT \* FROM (SELECT \* FROM product where typ = 'Pc') as prod LEFT SEMI JOIN pc ON prod.id = pc.id;

//Выдает только столбцы левой таблицы

**ACID ENABLED TABLE (TRANSACTION TABLE)**

В таблице с поддержкой “ACID” данные хранятся в виде набора базовых файлов. Новые записи, обновления и удаления сохраняются в виде дельта-файлов. Для каждой транзакции создается новый набор дельта-файлов.При считывании данных из таблицы считыватель смешивает базовые и дельта-записи, применяя любые обновления и удаления по мере чтения.

// так же должны быть включены флаги hive

Set [hive.enforce.bucketing](https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties#ConfigurationProperties-hive.enforce.bucketing)=true  
Set [hive.exec.dynamic.partition.mode](https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties#ConfigurationProperties-hive.exec.dynamic.partition.mode)= nonstrict  
Set h[ive.support.concurrency](https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties#ConfigurationProperties-hive.support.concurrency)=true

*ACID tables по-прежнему не поддерживает фиксацию и откат. Поддерживается только формат файла ORC. Таблицы должны быть объединены в группы, чтобы использовать эту функцию.*

CREATE TABLE IF NOT EXISTS orders (id bigint,  
customer\_id string,   
src\_update\_ts timestamp,)  
PARTITIONED BY (order\_date date)  
CLUSTERED BY (id) INTO 10 BUCKETS STORED AS ORC;  
TBLPROPERTIES ("transactional"="true")

**COMPACTION**

Решение проблемы с большим количеством мелких файлов, которые поступают в hdfs (например стримингом), или при частом обновлении таблиц, или большом количестве reducers

Например, перезапишем партиции которые имеют более 5-ти файлов:

with partition\_list as  
(select order\_date, count(distinct input\_\_file\_\_name) cnt from orders  
group by order\_date having cnt > 5)  
insert overwrite table orders partition (order\_date)  
select \* from orders  
where order\_date in (select order\_date from partition\_list)

**Spark-hive**

В SparkSession включить

enableHiveSupport()

Далее вам будут доступны таблицы hive как обычные view spark.

Что бы прочитать таблицу hive:

val df = spark.table("NameLogs")

Что бы добавить в таблицу:

df.write  
 .format("hive")  
 .insertInto("NameLogs")//.saveAsTable("NameLogs")