guisu，程序人生。 逆水行舟，不进则退。

能干的人解决问题。智慧的人绕开问题(A clever person solves a problem. A wise person avoids it)

个人资料

真实的归宿

访问： 3540011次
积分： 24746
等级：
排名： 第154名

原创： 212篇 转载： 2篇
译文： 0篇 评论： 1130条

Hadoop Hive sql语法详解

标签： hadoop sql insert table string

2012-02-14 09:37 83526人阅读 评论(6) 收藏 举报

本文章已收录于： Hadoop知识库

分类： 云计算hadoop（24）

Hive 是基于**Hadoop** 构建的一套数据仓库分析系统，它提供了丰富的SQL查询方式来分析存储在Hadoop分布式文件系统中的数据，可以将结构

化的数据文件映射为一张数据库表，并提供完整的SQL查询功能，可以将SQL语句转换为MapReduce任务进行运行，通过自己的SQL 去查询分析需

要的内容，这套SQL 简称Hive SQL，使不熟悉mapreduce 的用户很方便的利用SQL 语言查询，汇总，分析数据。而mapreduce开发人员可以把

己写的mapper 和reducer 作为插件来支持Hive 做更复杂的数据分析。

它与关系型数据库的SQL 略有不同，但支持了绝大多数的语句如DDL、DML 以及常见的聚合函数、连接查询、条件查询。HIVE不适合用于联机

online)事务处理，也不提供实时查询功能。它最适合应用在基于大量不可变数据的批处理作业。

HIVE的特点：可伸缩（在Hadoop的集群上动态的添加设备），可扩展，容错，输入格式的松散耦合。

Hive 的官方文档中对查询语言有了很详细的描述，请参考：
http://wiki.apache.org/hadoop/Hive/LanguageManual ，本文的内容大部分翻译自该页面，期间加入了一些在使用过程中需要注意到的事项。

1. DDL 操作

DDL

- 建表
- 删除表
- 修改表结构
- 创建 / 删除视图
- 创建数据库
- 显示命令

建表：

```
CREATE [EXTERNAL] TABLE [IF NOT EXISTS] table_name
  [(col_name data_type [COMMENT col_comment], ...)]
  [COMMENT table_comment]
  [PARTITIONED BY (col_name data_type [COMMENT col_comment], ...)]
  [CLUSTERED BY (col_name, col_name, ...)
  [SORTED BY (col_name [ASC|DESC], ...)] INTO num_buckets BUCKETS]
  [ROW FORMAT row_format]
```

关于php的libevent扩展的应用 (0)
libevent简单介绍 (0)
SOA (0)

文章存档
2015年12月 (2)
2015年10月 (4)
2015年05月 (2)
2015年04月 (1)
2015年01月 (2)

阅读排行
八大排序算法 (300743)
Mysql 多表联合查询效率 (146062)
深入理解java异常处理机 (115673)
socket阻塞与非阻塞，同 (107014)
hbase安装配置（整合到 (92065)
设计模式 (十八) 策略模: (90112)
Linux的SOCKET编程详 (87670)
Hadoop Hive sql语法详 (83333)
Nginx工作原理和优化、 (79749)
Java输入输出流 (74798)

评论排行
八大排序算法 (100)
深入理解java异常处理机 (94)
socket阻塞与非阻塞，同 (53)
硬盘的读写原理 (41)
设计模式 (十八) 策略模: (41)
设计模式（一）工厂模式 (32)
海量数据处理算法—Bit-M (26)
UML图中类之间的关系:((23)
Redis应用场景 (23)
PHP SOCKET编程 (23)

推荐文章
*RxJava----操作符：组合Observable
*Binder工作机制
* Java Web基础知识之Filter：过滤一切你不想看到的事情/a>
*Untiy Native Render Plugin在VR中的绘制(二): 透明排序
*随机过程--Metropolis-Hastings算法
*Fresco图片库研读分析

最新评论
八大排序算法
shuai1210: @qq_18115123:应该先判断索引 j>=0
socket阻塞与非阻塞，同步与异
一个人的场域: GOOD！
设计模式概论

[STORED AS file_format]
[LOCATION hdfs_path]

•CREATE TABLE 创建一个指定名字的表。如果相同名字的表已经存在，则抛出异常；用户可以用 IF NOT EXIST 选项来忽略这个异常
•EXTERNAL 关键字可以让用户创建一个外部表，在建表的同时指定一个指向实际数据的路径（LOCATION）
•LIKE 允许用户复制现有的表结构，但是不复制数据
•COMMENT可以为表与字段增加描述

•ROW FORMAT
DELIMITED [FIELDS TERMINATED BY char] [COLLECTION ITEMS TERMINATED BY char]
[MAP KEYS TERMINATED BY char] [LINES TERMINATED BY char]
| SERDE serde_name [WITH SERDEPROPERTIES (property_name=property_value, property_name=property_value, ...)]

用户在建表的时候可以自定义 SerDe 或者使用自带的 SerDe。如果没有指定 ROW FORMAT 或者 ROW FORMAT DELIMITED，将会使用自带的 SerDe。在建表的时候，用户还需要为表指定列，用户在指定表的列的同时也会指定自定义的 SerDe，Hive 通过 SerDe 确定表的具体的列的数据。

•STORED AS
SEQUENCEFILE
| TEXTFILE
| RCFILE
| INPUTFORMAT input_format_classname OUTPUTFORMAT output_format_classname

如果文件数据是纯文本，可以使用 STORED AS TEXTFILE。如果数据需要压缩，使用 STORED AS SEQUENCE 。

创建简单表：

```
hive> CREATE TABLE pokes (foo INT, bar STRING);
```

创建外部表：

```
CREATE EXTERNAL TABLE page_view(viewTime INT, userid BIGINT,
    page_url STRING, referrer_url STRING,
    ip STRING COMMENT 'IP Address of the User',
    country STRING COMMENT 'country of origination')
 COMMENT 'This is the staging page view table'
 ROW FORMAT DELIMITED FIELDS TERMINATED BY '\054'
 STORED AS TEXTFILE
 LOCATION '<hdfs_location>';
```

建分区表

```
CREATE TABLE par_table(viewTime INT, userid BIGINT,
    page_url STRING, referrer_url STRING,
    ip STRING COMMENT 'IP Address of the User')
 COMMENT 'This is the page view table'
 PARTITIONED BY(date STRING, pos STRING)
ROW FORMAT DELIMITED '\t'
  FIELDS TERMINATED BY '\n'
STORED AS SEQUENCEFILE;
```

建**Bucket**表

```
CREATE TABLE par_table(viewTime INT, userid BIGINT,
    page_url STRING, referrer_url STRING,
    ip STRING COMMENT 'IP Address of the User')
 COMMENT 'This is the page view table'
 PARTITIONED BY(date STRING, pos STRING)
 CLUSTERED BY(userid) SORTED BY(viewTime) INTO 32 BUCKETS
 ROW FORMAT DELIMITED '\t'
  FIELDS TERMINATED BY '\n'
STORED AS SEQUENCEFILE;
```

创建表并创建索引字段**ds**

```
hive> CREATE TABLE invites (foo INT, bar STRING) PARTITIONED BY (ds STRING);
```

复制一个空表

```
CREATE TABLE empty_key_value_store
```

LIKE key_value_store;

例子

```
create table  user_info (user_id int, cid string, ckid string, username string)

row format delimited

fields terminated by '\t'

 lines terminated by '\n';
```

导入数据表的数据格式是：字段之间是tab键分割，行之间是断行。

及要我们的文件内容格式：

```
100636  100890  c5c86f4cddc15eb7        yyyvybtvt
100612  100865  97cc70d411c18b6f        gyvcycy
100078  100087  ecd6026a15ffddf5        qa000100
```

显示所有表：

```
hive> SHOW TABLES;
```

按正条件（正则表达式）显示表，

```
hive> SHOW TABLES '.*s';
```

修改表结构

- 增加分区、删除分区
- 重命名表
- 修改列的名字、类型、位置、注释
- 增加/更新列
- 增加表的元数据信息

表添加一列 ：

```
hive> ALTER TABLE pokes ADD COLUMNS (new_col INT);
```

添加一列并增加列字段注释

```
hive> ALTER TABLE invites ADD COLUMNS (new_col2 INT COMMENT 'a comment');
```

更改表名：

```
hive> ALTER TABLE events RENAME TO 3koobecaf;
```

删除列：

```
hive> DROP TABLE pokes;
```

增加、删除分区

- 增加

```
ALTER TABLE table_name ADD [IF NOT EXISTS] partition_spec [ LOCATION 'location1' ]
partition_spec [ LOCATION 'location2' ] ...
     partition_spec:
  : PARTITION (partition_col = partition_col_value, partition_col = partiton_col_value, ...)
```

- 删除

```
ALTER TABLE table_name DROP partition_spec, partition_spec,...
```

重命名表

- ALTER TABLE table_name RENAME TO new_table_name

修改列的名字、类型、位置、注释：

- ALTER TABLE table_name CHANGE [COLUMN] col_old_name col_new_name column_type

[COMMENT col_comment] [FIRST|AFTER column_name]
•这个命令可以允许改变列名、数据类型、注释、列位置或者它们的任意组合
表添加一列 ：

hive> ALTER TABLE pokes ADD COLUMNS (new_col INT);

添加一列并增加列字段注释

hive> ALTER TABLE invites ADD COLUMNS (new_col2 INT COMMENT 'a comment');

增加/更新列
•ALTER TABLE table_name ADD|REPLACE COLUMNS (col_name data_type [COMMENT col_comment], ...)

• ADD是代表新增一字段，字段位置在所有列后面(partition列前)
REPLACE则是表示替换表中所有字段。
增加表的元数据信息
•ALTER TABLE table_name SET TBLPROPERTIES table_properties table_properties:
:[property_name = property_value.....]

•用户可以用这个命令向表中增加metadata
改变表文件格式与组织
•ALTER TABLE table_name SET FILEFORMAT file_format
•ALTER TABLE table_name CLUSTERED BY(userid) SORTED BY(viewTime) INTO num_buckets BUCKETS

•这个命令修改了表的物理存储属性
创建 / 删除视图
•CREATE VIEW [IF NOT EXISTS] view_name [(column_name [COMMENT column_comment], ...)] [COMMENT view_comment][TBLPROPERTIES (property_name = property_value, ...)] AS SELECT
•增加视图
•如果没有提供表名，视图列的名字将由定义的SELECT表达式自动生成
•如果修改基本表的属性，视图中不会体现，无效查询将会失败
•视图是只读的，不能用LOAD/INSERT/ALTER
•DROP VIEW view_name
•删除视图
创建数据库
•CREATE DATABASE name
显示命令
•show tables;
•show databases;
•show partitions ;
•show functions
•describe extended table_name dot col_name

2. DML 操作:元数据存储

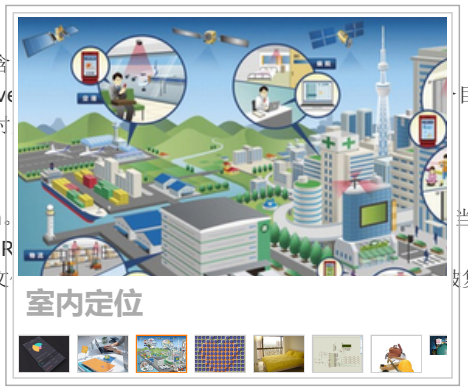
hive不支持用insert语句一条一条的进行插入操作，也不支持update操作。数据是以load的方式加载到建立好的表中。数据一旦导入就不可以修改。

DML包括：INSERT插入、UPDATE更新、DELETE删除

•向数据表内加载文件
•将查询结果插入到Hive表中
•0.8新特性 insert into

向数据表内加载文件
•LOAD DATA [LOCAL] INPATH 'filepath' [OVERWRITE] INTO TABLE tablename [PARTITION (partcol1=val1, partcol2=val2 ...)]
•Load 操作只是单纯的复制/移动操作，将数据文件移动到 Hive 表对应的位置。
•filepath
•相对路径，例如：project/data1
•绝对路径，例如： /user/hive/project/data1
•包含模式的完整 URI，例如：hdfs://namenode:9000/user/hive/project/data1
例如：

hive> LOAD DATA LOCAL INPATH './examples/files/kv1.txt' OVERWRITE INTO TABLE pokes;



加载本地数据，同时给定分区信息

•加载的目标可以是一个表或者分区。如果表包含

•filepath 可以引用一个文件（这种情况下，Hive 目录（在这种情况下，Hive 会将目录中的所有文件移动至表所对

LOCAL关键字

•指定了LOCAL，即本地

•load 命令会去查找本地文件系统中的 filepath。 当前用户的当前路径。用户也可以为本地文件指定一个完整的 UR

•load 命令会将 filepath 中的文件复制到目标文 复制的数据文件移动到表的数据对应的位置

例如：加载本地数据，同时给定分区信息：

hive> LOAD DATA LOCAL INPATH './examples/files/kv2.txt' OVERWRITE INTO TABLE invites PARTITION (ds='2008-08-15');

• 没有指定LOCAL

如果 filepath 指向的是一个完整的 URI，hive 会直接使用这个 URI。 否则

•如果没有指定 schema 或者 authority，Hive 会使用在 hadoop 配置文件中定义的 schema 和 authority，fs.default.name 指定了 Namenode 的 URI

•如果路径不是绝对的，Hive 相对于 /user/ 进行解释。 Hive 会将 filepath 中指定的文件内容移动到 table （或者 partition）所指定的路径中

加载**DFS**数据 ，同时给定分区信息：

hive> LOAD DATA INPATH '/user/myname/kv2.txt' OVERWRITE INTO TABLE invites PARTITION (ds='2008-08-15');

The above command will load data from an HDFS file/directory to the table. Note that loading data from HDFS will result in moving the file/directory. As a result, the operation is almost instantaneous.

OVERWRITE

•指定了OVERWRITE

•目标表（或者分区）中的内容（如果有）会被删除，然后再将 filepath 指向的文件/目录中的内容添加到表/分区中。

•如果目标表（分区）已经有一个文件，并且文件名和 filepath 中的文件名冲突，那么现有的文件会被新文件所替代。

将查询结果插入**Hive**表

•将查询结果插入Hive表

•将查询结果写入HDFS文件系统

•基本模式

INSERT OVERWRITE TABLE tablename1 [PARTITION (partcol1=val1, partcol2=val2 ...)] select_statement1 FROM from_statement

•多插入模式

FROM from_statement

INSERT OVERWRITE TABLE tablename1 [PARTITION (partcol1=val1, partcol2=val2 ...)] select_statement1

[INSERT OVERWRITE TABLE tablename2 [PARTITION ...] select_statement2] ...

•自动分区模式

INSERT OVERWRITE TABLE tablename PARTITION (partcol1[=val1], partcol2[=val2] ...) select_statement FROM from_statement

将查询结果写入**HDFS**文件系统

•INSERT OVERWRITE [LOCAL] DIRECTORY directory1 SELECT ... FROM ...

FROM from_statement

INSERT OVERWRITE [LOCAL] DIRECTORY directory1 select_statement1

[INSERT OVERWRITE [LOCAL] DIRECTORY directory2 select_statement2]

•

•数据写入文件系统时进行文本序列化，且每列用^A 来区分，\n换行

INSERT INTO

•INSERT INTO TABLE tablename1 [PARTITION (partcol1=val1, partcol2=val2 ...)] select_statement1 FROM from_statement

3. DQL 操作:数据查询SQL

SQL操作
•基本的Select 操作
•基于Partition的查询
•Join

3.1 基本的Select 操作

```
SELECT [ALL | DISTINCT] select_expr, select_expr, ...
FROM table_reference
[WHERE where_condition]
[GROUP BY col_list [HAVING condition]]
[   CLUSTER BY col_list
  | [DISTRIBUTE BY col_list] [SORT BY| ORDER BY col_list]
]
[LIMIT number]
```

•使用ALL和DISTINCT选项区分对重复记录的处理。默认是ALL，表示查询所有记录。DISTINCT表示去掉重复的记录
•
•Where 条件
•类似我们传统SQL的where 条件
•目前支持 AND,OR ,0.9版本支持between
•IN, NOT IN
•不支持EXIST ,NOT EXIST

ORDER BY与**SORT BY**的不同
•ORDER BY 全局排序，只有一个Reduce任务
•SORT BY 只在本机做排序

Limit
•Limit 可以限制查询的记录数

```
SELECT * FROM t1 LIMIT 5
```

•实现Top k 查询
•下面的查询语句查询销售记录最大的 5 个销售代表。

```
SET mapred.reduce.tasks = 1
  SELECT * FROM test SORT BY amount DESC LIMIT 5
```

•REGEX Column Specification
SELECT 语句可以使用正则表达式做列选择，下面的语句查询除了 ds 和 hr 之外的所有列：

```
SELECT `(ds|hr)?+.+` FROM test
```

例如
按先件查询

```
hive> SELECT a.foo FROM invites a WHERE a.ds='<DATE>';
```

将查询数据输出至目录：

```
hive> INSERT OVERWRITE DIRECTORY '/tmp/hdfs_out' SELECT a.* FROM invites a WHERE a.ds='<DATE>';
```

将查询结果输出至本地目录：

```
hive> INSERT OVERWRITE LOCAL DIRECTORY '/tmp/local_out' SELECT a.* FROM pokes a;
```

选择所有列到本地目录：

```
hive> INSERT OVERWRITE TABLE events SELECT a.* FROM profiles a;
hive> INSERT OVERWRITE TABLE events SELECT a.* FROM profiles a WHERE a.key < 100;
hive> INSERT OVERWRITE LOCAL DIRECTORY '/tmp/reg_3' SELECT a.* FROM events a;
hive> INSERT OVERWRITE DIRECTORY '/tmp/reg_4' select a.invites, a.pokes FROM profiles a;
hive> INSERT OVERWRITE DIRECTORY '/tmp/reg_5' SELECT COUNT(1) FROM invites a WHERE
a.ds='<DATE>';
hive> INSERT OVERWRITE DIRECTORY '/tmp/reg_5' SELECT a.foo, a.bar FROM invites a;
hive> INSERT OVERWRITE LOCAL DIRECTORY '/tmp/sum' SELECT SUM(a.pc) FROM pc1 a;
```

将一个表的统计结果插入另一个表中：

```
hive> FROM invites a INSERT OVERWRITE TABLE events SELECT a.bar, count(1) WHERE a.foo > 0 GROUP BY a.bar;
hive> INSERT OVERWRITE TABLE events SELECT a.bar, count(1) FROM invites a WHERE a.foo > 0 GROUP BY a.bar;
```

JOIN

```
hive> FROM pokes t1 JOIN invites t2 ON (t1.bar = t2.bar) INSERT OVERWRITE TABLE events SELECT t1.bar, t1.foo, t2.foo;
```

将多表数据插入到同一表中：

```
FROM src
INSERT OVERWRITE TABLE dest1 SELECT src.* WHERE src.key < 100
INSERT OVERWRITE TABLE dest2 SELECT src.key, src.value WHERE src.key >= 100 and src.key < 200
INSERT OVERWRITE TABLE dest3 PARTITION(ds='2008-04-08', hr='12') SELECT src.key WHERE src.key >= 200 and src.key < 300
INSERT OVERWRITE LOCAL DIRECTORY '/tmp/dest4.out' SELECT src.value WHERE src.key >= 300;
```

将文件流直接插入文件：

```
hive> FROM invites a INSERT OVERWRITE TABLE events SELECT TRANSFORM(a.foo, a.bar) AS (oof, rab) USING '/bin/cat' WHERE a.ds > '2008-08-09';
```

This streams the data in the map phase through the script /bin/cat (like hadoop streaming). Similarly - streaming can be used on the reduce side (please see the Hive Tutorial or examples)

3.2 基于**Partition**的查询

•一般 SELECT 查询会扫描整个表，使用 PARTITIONED BY 子句建表，查询就可以利用分区剪枝（input pruning）的特性
•Hive 当前的实现是，只有分区断言出现在离 FROM 子句最近的那个WHERE 子句中，才会启用分区剪枝

3.3 Join

Syntax

```
join_table:
   table_reference JOIN table_factor [join_condition]
  | table_reference {LEFT|RIGHT|FULL} [OUTER] JOIN table_reference join_condition
  | table_reference LEFT SEMI JOIN table_reference join_condition

table_reference:
    table_factor
  | join_table

table_factor:
    tbl_name [alias]
  | table_subquery alias
  | ( table_references )

join_condition:
    ON equality_expression ( AND equality_expression )*

equality_expression:
    expression = expression
```

•Hive 只支持等值连接（equality joins）、外连接（outer joins）和（left semi joins）。Hive 不支持所有非等值的连接，因为非等值连接非常难转化到 map/reduce 任务

•LEFT，RIGHT和FULL OUTER关键字用于处理join中空记录的情况
•LEFT SEMI JOIN 是 IN/EXISTS 子查询的一种更高效的实现
•join 时，每次 map/reduce 任务的逻辑是这样的：reducer 会缓存 join 序列中除了最后一个表的所有表的记录，再通过最后一个表将结果序列化到文件系统
•实践中，应该把最大的那个表写在最后
join 查询时，需要注意几个关键点

•只支持等值join
•SELECT a.* FROM a JOIN b ON (a.id = b.id)
•SELECT a.* FROM a JOIN b
ON (a.id = b.id AND a.department = b.department)
•可以 join 多于 2 个表，例如
SELECT a.val, b.val, c.val FROM a JOIN b
ON (a.key = b.key1) JOIN c ON (c.key = b.key2)

•如果join中多个表的 join key 是同一个，则 join 会被转化为单个 map/reduce 任务
LEFT，RIGHT和FULL OUTER
•例子
•SELECT a.val, b.val FROM a LEFT OUTER JOIN b ON (a.key=b.key)

•如果你想限制 join 的输出，应该在 WHERE 子句中写过滤条件——或是在 join 子句中写
•
•容易混淆的问题是表分区的情况
• SELECT c.val, d.val FROM c LEFT OUTER JOIN d ON (c.key=d.key)
WHERE a.ds='2010-07-07' AND b.ds='2010-07-07'
•如果 d 表中找不到对应 c 表的记录，d 表的所有列都会列出 NULL，包括 ds 列。也就是说，join 会过滤 d 表中不能找到匹配 c 表 join key 的所有记录。这样的话，LEFT OUTER 就使得查询结果与 WHERE 子句无关
•解决办法
•SELECT c.val, d.val FROM c LEFT OUTER JOIN d
ON (c.key=d.key AND d.ds='2009-07-07' AND c.ds='2009-07-07')
LEFT SEMI JOIN
•LEFT SEMI JOIN 的限制是， JOIN 子句中右边的表只能在 ON 子句中设置过滤条件，在 WHERE 子句、SELECT 子句或其他地方过滤都不行
•
•SELECT a.key, a.value
FROM a
WHERE a.key in
(SELECT b.key
FROM B);
可以被重写为：
SELECT a.key, a.val
FROM a LEFT SEMI JOIN b on (a.key = b.key)
UNION ALL
•用来合并多个select的查询结果，需要保证select中字段须一致
•select_statement UNION ALL select_statement UNION ALL select_statement ...

4. 从SQL到HiveQL应转变的习惯

1、Hive不支持等值连接
•SQL中对两表内联可以写成：
•select * from dual a,dual b where a.key = b.key;
•Hive中应为
•select * from dual a join dual b on a.key = b.key;
而不是传统的格式：

```
SELECT t1.a1 as c1, t2.b1 as c2FROM t1, t2
 WHERE t1.a2 = t2.b2
```

2、分号字符
•分号是SQL语句结束标记，在HiveQL中也是，但是在HiveQL中，对分号的识别没有那么智慧，例如：
•select concat(key,concat(';',key)) from dual;
•但HiveQL在解析语句时提示：
FAILED: Parse Error: line 0:-1 mismatched input '<EOF>' expecting) in function specification
•解决的办法是，使用分号的八进制的ASCII码进行转义，那么上述语句应写成：
•select concat(key,concat('\073',key)) from dual;

3、IS [NOT] NULL
•SQL中null代表空值, 值得警惕的是, 在HiveQL中String类型的字段若是空(empty)字符串, 即长度为0, 那么对它进行IS NULL的判断结果是False.
4、Hive不支持将数据插入现有的表或分区中，

仅支持覆盖重写整个表，示例如下：

```
[sql] view plain copy print ?
INSERT OVERWRITE TABLE t1
SELECT * FROM t2;
```

4、hive不支持INSERT INTO, UPDATE, DELETE操作

这样的话，就不要很复杂的锁机制来读写数据。

INSERT INTO syntax is only available starting in version 0.8。INSERT INTO就是在表或分区中追加数据。

5、hive支持嵌入mapreduce程序，来处理复杂的逻辑

如：

```
[sql] view plain copy print ?
FROM (
MAP doctext USING 'python wc_mapper.py' AS (word, cnt)
FROM docs
CLUSTER BY word
) a
REDUCE word, cnt USING 'python wc_reduce.py';
```

--doctext: 是输入

--word, cnt: 是map程序的输出

--CLUSTER BY: 将wordhash后，又作为reduce程序的输入

并且map程序、reduce程序可以单独使用，如：

```
[sql] view plain copy print ?
FROM (
FROM session_table
SELECT sessionid, tstamp, data
DISTRIBUTE BY sessionid SORT BY tstamp
) a
REDUCE sessionid, tstamp, data USING 'session_reducer.sh';
```

--DISTRIBUTE BY: 用于给reduce程序分配行数据

6、hive支持将转换后的数据直接写入不同的表，还能写入分区、hdfs和本地目录。

这样能免除多次扫描输入表的开销。

```
[sql] view plain copy print ?
FROM t1

INSERT OVERWRITE TABLE t2
SELECT t3.c2, count(1)
FROM t3
WHERE t3.c1 <= 20
GROUP BY t3.c2

INSERT OVERWRITE DIRECTORY '/output_dir'
SELECT t3.c2, avg(t3.c1)
FROM t3
WHERE t3.c1 > 20 AND t3.c1 <= 30
GROUP BY t3.c2

INSERT OVERWRITE LOCAL DIRECTORY '/home/dir'
SELECT t3.c2, sum(t3.c1)
FROM t3
WHERE t3.c1 > 30
GROUP BY t3.c2;
```

5. 实际示例

创建一个表

CREATE TABLE u_data (

```
userid INT,
movieid INT,
rating INT,
unixtime STRING)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '/t'
STORED AS TEXTFILE;
```

下载示例数据文件，并解压缩

```
wget http://www.grouplens.org/system/files/ml-data.tar__0.gz
tar xvzf ml-data.tar__0.gz
```

加载数据到表中：

```
LOAD DATA LOCAL INPATH 'ml-data/u.data'
OVERWRITE INTO TABLE u_data;
```

统计数据总量：

```
SELECT COUNT(1) FROM u_data;
```

现在做一些复杂的数据分析：

创建一个 weekday_mapper.py: 文件，作为数据按周进行分割

```
import sys
import datetime

for line in sys.stdin:
line = line.strip()
userid, movieid, rating, unixtime = line.split('/t')
```

生成数据的周信息

```
weekday = datetime.datetime.fromtimestamp(float(unixtime)).isoweekday()
print '/t'.join([userid, movieid, rating, str(weekday)])
```

使用映射脚本

```
//创建表，按分割符分割行中的字段值
CREATE TABLE u_data_new (
userid INT,
movieid INT,
rating INT,
weekday INT)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '/t';
//将Python文件加载到系统
add FILE weekday_mapper.py;
```

将数据按周进行分割

```
INSERT OVERWRITE TABLE u_data_new
SELECT
TRANSFORM (userid, movieid, rating, unixtime)
USING 'python weekday_mapper.py'
AS (userid, movieid, rating, weekday)
FROM u_data;

SELECT weekday, COUNT(1)
FROM u_data_new
```

```
GROUP BY weekday;
```

处理**Apache Weblog** 数据

将WEB日志先用正则表达式进行组合，再按需要的条件进行组合输入到表中

```
add jar ../build/contrib/hive_contrib.jar;

CREATE TABLE apachelog (
host STRING,
identity STRING,
user STRING,
time STRING,
request STRING,
status STRING,
size STRING,
referer STRING,
agent STRING)
ROW FORMAT SERDE 'org.apache.hadoop.hive.contrib.serde2.RegexSerDe'
WITH SERDEPROPERTIES (
"input.regex" = "([^ ]*) ([^ ]*) ([^ ]*) (-|[^/][^/]]*//]) ([^ /"]*|/"[^/"]*/") (-|[0-9]*) (-|[0-9]*)(?: ([^ /"]*|/"[^/"]*/") ([^ /"]*|/"
[^/"]*/"))?",
"output.format.string" = "%1$s %2$s %3$s %4$s %5$s %6$s %7$s %8$s %9$s"
)
STORED AS TEXTFILE;
```


上一篇 网络互联参考模型（详解）
下一篇 HDFS写入和读取流程

我的同类文章

云计算**hadoop**（**24**）

- 跟上节奏 大数据时代十大必… 2015-05-30 阅读 2067
- 深入解析：分布式系统的事务.. 2014-01-21 阅读 6717
- 你的数据根本不够大，别老扯.. 2013-10-10 阅读 6378
- 海量数据处理算法—Bloom F… 2012-08-14 阅读 26663
- PHP通过Thrift操作Hbase 2012-02-27 阅读 17098
- Hive深入浅出 2014-02-19 阅读 7386
- 数据分析≠Hadoop+NoSQL… 2013-12-23 阅读 4774
- Storm集群安装详解 2013-07-25 阅读 12404
- 海量数据处理 2012-08-11 阅读 4232
- Hadoop Hive与Hbase整合+… 2012-02-22 阅读 31709

更多文章

参考知识库

猜你在找

查看评论

6楼 Nicolas_Chan 2016-04-21 13:43发表

不错，学习了

5楼 lifangzhen0 2015-07-15 15:56发表

hiveSQL入门，写的很好

4楼 siki5201314 2014-09-19 13:56发表

good

3楼 To-Big_Fish 2014-04-29 16:29发表

nice

2楼 wangxiaojing123 2013-01-15 16:21发表

写的好，谢谢分享。

1楼 xxzapple 2012-02-14 17:16发表

写的好！