Hive的数据类型 - link

数值类型：
tinyint/smallint/int/bigint：整数类型
float/double/decimal：浮点数类型

字符串类型：
string/varchar/char：字符串类型（varchar长度不固定，char长度固定）

时间类型：
timestamp/date

时间操作：
将时间戳(bigint)转为日期时间的字符串(string)，timestamp单位为毫秒：
from_unixtime(floor(`timestamp`/1000))
或者指定格式：from_unixtime(floor(`timestamp`/1000), "yyyy-MM-dd")，格式为："yyyy-MM-dd HH:mm:ss"

其他类型：
boolean：布尔类型
binary：二进制类型

复杂数据类型：
arrays: ARRAY<data_type>
maps: MAP<primitive_type, data_type>
structs: STRUCT<col_name : data_type [COMMENT col_comment], ...>
union: UNIONTYPE<data_type, data_type, ...>：

array的使用
hive> create table tbl_name (a array<string>, b array<string>)
ROW FORMAT
DELIMITED FIELDS TERMINATED BY '\t'
COLLECTION ITEMS TERMINATED BY ',';

hive> load data local inpath "../hive/examples/files/arraytest.txt" overwrite into table tbl_name;
b00,b01 b00,b01
b00,b01 b00,b01
b00,b01 b00,b01
b00,b01 b00,b01

hive> select a from tbl_name;
hive> select a[0] from tbl_name;

函数：
array_contains(Array<T>, value)：Returns TRUE if the array contains value.

array(n0, n1...)：该函数用于生成一个array，可使用 desc function array; 来查看该函数的使用，其中n0，n1可为表中字段。
collect_list(Returns a list of objects with duplicates)函数可以将一列数据转化为一个array，在业务逻辑中更新active_watch时会应用这个函数创建event字段所需的array，但是需要加distinct；另外一个函数collect_set(Returns a set of objects with duplicate elements eliminated)则是返回一个set。

map的使用
CREATE TABLE tbl_name (foo STRING , bar MAP<STRING, STRING>)
ROW FORMAT
DELIMITED FIELDS TERMINATED BY '\t'
COLLECTION ITEMS TERMINATED BY ','
MAP KEYS TERMINATED BY ':'
STORED AS TEXTFILE;

hive> load data local inpath "../hive/examples/files/maptest.txt" overwrite into table tbl_name;
a00	b0:b01,b1:b11
a01	b1:b11,b2:b12
a02	b2:b12,b3:b13
a03	b3:b13,b4:b14

hive> select bar from tbl_name;
hive> select bar[‘b1’] from tbl_name;
hive> select size(bar) from tbl_name;

Hive的表操作 - link

内部表：
examples：
hive> create table t1(tid int, tname string, age int);
hive> create table t2(tid int, tname string, age int)
    > location 'mytable/hive/t2';
hive> create table t3(tid int, tname string, age int)
    > row format delimited fields terminated by ',';
hive> create table t4 as select * from sample_data;

导入数据：LOAD DATA [LOCAL] INPATH 'filepath' [OVERWRITE] INTO TABLE tablename [PARTITION (partcol1=val1, partcol2=val2 ...)]

分区表：
example：
hive> create table partition_table(sid int, sname string)
    > partitioned by (gender string);
hive> insert into table partition_table partition(gender=’M’) select sid,sname from sample_data where gender=’M’;

hive> insert into table partition_table partition(gender='F') select sid,sname from sample_data where gender='F';

外部表:
example:
hive> create external table exter_table(sid int, sname string)
    > location '/user/hive/external';
删除表时hdfs中文件不会删除，仅删除元数据。

桶表:
对其中一列进行hash后存储。
example:
hive> create table bucket_table(sid int, sname string)
    > clustered by(sname) into 5 buckets;

视图:
是一个虚表，建立在基表之上。
example:
hive> create view empinto
    > as select e.empno, e.ename, e.sal, e.sal*12 annlsal, d.dname
    > from emp e, dept d where e.deptno=d.deptno;

删除表:
drop table table_name;
drop table if exists table_name;

修改表:
ALTER TABLE name RENAME TO new_name
ALTER TABLE name ADD COLUMNS (col_spec[, col_spec ...])
ALTER TABLE name DROP [COLUMN] column_name
ALTER TABLE name CHANGE column_name new_name new_type
ALTER TABLE name REPLACE COLUMNS (col_spec[, col_spec ...])

插入表:
INSERT OVERWRITE TABLE tablename1 [PARTITION (partcol1=val1, partcol2=val2 ...) [IF NOT EXISTS]] select_statement1 FROM from_statement;
INSERT INTO TABLE tablename1 [PARTITION (partcol1=val1, partcol2=val2 ...)] select_statement1 FROM from_statement;

删除表中数据:
truncate table tbl_name

累加
原表(如果某一天有多个记录可以用group by将某一天的所有记录group为一条):test(time string, num int)
2016-02-23 1

2016-02-24 2
累加后的表:
2016-02-23 1
2016-02-24 3

hive> select time, sum(num) over (order by time) from test;

创建分区表

Add partitions:

```
ALTER TABLE table_name ADD [IF NOT EXISTS] PARTITION partition_spec
  [LOCATION 'location1'] partition_spec [LOCATION 'location2'] ...;
```

应用商店浏览和下载数据的一个例子:

```
create external table app_log_14(json string) partitioned by (`date` string);
alter table app_log_14 add partition (`date` = '2016-02') location
'/app_log/ali-hz-search-14/2016/02';
select count(*) from app_log_14 where `date` = '2016-02';
```

浏览数据表:

```
create table app_detail_log(`from` string, package_name string, query string, device_id
string, message_type int, remote_ip string, time bigint, uid string) partitioned by (`date`
string);

insert into table app_detail_log partition(`date` = '2016-02') select
get_json_object(app_log_14.json, '$.detail_log.from'), get_json_object(app_log_14.json,
'$.detail_log.package_name'), get_json_object(app_log_14.json, '$.detail_log.query'),
get_json_object(app_log_14.json, '$.device_id'), get_json_object(app_log_14.json,
'$.message_type'), get_json_object(app_log_14.json, '$.remote_ip'),
get_json_object(app_log_14.json, '$.time'), get_json_object(app_log_14.json, '$.uid') from
app_log_14 where get_json_object(app_log_14.json, '$.detail_log.package_name') is not
NULL and `date` = '2016-02';
```

下载数据表:

```
create table app_download_log(device_id string, apk_url string, appid int, message_type int,
remote_ip string, time bigint, uid string) partitioned by (`date` string);

insert into table app_download_log partition(`date` = '2016-02') select
get_json_object(app_log_14.json, '$.device_id'), get_json_object(app_log_14.json,
'$.download_log.apk_url'), get_json_object(app_log_14.json, '$.download_log.appid'),
get_json_object(app_log_14.json, '$.message_type'), get_json_object(app_log_14.json,
'$.remote_ip'), get_json_object(app_log_14.json, '$.time'), get_json_object(app_log_14.json,
'$.uid') from app_log_14 where get_json_object(app_log_14.json, '$.download_log.apk_url')
is not NULL and `date` = '2016-02';
```

Rename partition:

ALTER TABLE table_name PARTITION partition_spec RENAME TO PARTITION partition_spec;

Exchange partition:
ALTER TABLE table_name_1 EXCHANGE PARTITION (partition_spec) WITH TABLE table_name_2;

Drop partitions:
ALTER TABLE table_name DROP [IF EXISTS] PARTITION partition_spec[, PARTITION partition_spec, ...]
  [IGNORE PROTECTION] [PURGE];

动态分区表 - link

// hive.exec.dynamic.partition needs to be set to true to enable dynamic partitioning with ALTER PARTITION
SET hive.exec.dynamic.partition = true;
或者在配置文件中设置。

所有列动态分区:
必须设置为nonstrict模式:set hive.exec.dynamic.partition.mode=nonstrict
INSERT OVERWRITE TABLE T PARTITION (ds, hr) SELECT key, value, ds, hr FROM srcpart WHERE ds is not null and hr>10;

部分列动态分区:
INSERT OVERWRITE TABLE T PARTITION (ds='2010-03-03', hr) SELECT key, value, /*ds,*/ hr FROM srcpart WHERE ds is not null and hr>10;

多表插入:
FROM SINSERT OVERWRITE TABLE T PARTITION (ds='2010-03-03', hr) SELECT key, value, ds, hr FROM srcpart WHERE ds is not null and hr>10INSERT OVERWRITE TABLE R PARTITION (ds='2010-03-03, hr=12)SELECT key, value, ds, hr from srcpart where ds is not null and hr = 12;

最大分区数设置:
set hive.exec.max.dynamic.partitions = 10000;
set hive.exec.max.dynamic.partitions.pernode = 1000;

字符串操作

分割字符串:**split**
split(string str, string pat)
按照pat字符串分割str，会返回分割后的字符串数组

高级查询

count
count(*) 所有值不全为NULL时，加1操作
count(1) 不管有没有值，只要有这条记录，值就加1

count(col) col列里面的值为NULL，值不会加1，这个列里面的值不为NULL，才加1

避免使用**count(distinct())**
使用group by代替，以下两句功能一样，grouop by速度比count distinct快：
select count(distinct user) from some_table;
select count(*) from (select user from some_table group by user) q;

sum 求和

avg 求平均

distinct 去重

like 模糊查询 %

order by 排序
select col1,other...
from table
where conditio
order by col1,col2 [asc|desc]
order by只有一个reducer，速度慢

sort by
sort by只是确保每个reduce上输出的数据有序，如果只有一个reduce时，和order by作用一样

distribute by
按照指定的字段对数据进行划分到不同的输出reduce文件中，一般和sort by配合使用

cluster by
把有相同值的数据聚集到一起，并排序。效果等价于distribute by col sort by col

group by
按照某些字段的值进行分组，有相同值放到一起
select col1 [,col2] ,count(col), count(distinct(col)), sel_expr（等聚合操作）from table
where condition -->Map端执行
group by col1 [,col2] -->Reduce端执行
[having] -->Reduce端执行

having
having对分组后的结果再次过滤，使用having可以避免group by后的子查询：

```
hive> SELECT year(ymd),avg(price_close) FROM stocks
    > WHERE exchange = 'NASDAQ' AND symbol = 'AAPL'
    > GROUP BY year(ymd)
    > HAVING avg(price_close) > 50.0;
1987  53.88968399108163
```

```
1991  52.49553383386182
1992  54.80338610251119
…
```

如果不使用having，那就需要使用子查询：

```
hive> SELECT s2.year,s2.avg FROM
        > (SELECT year(ymd) AS year,avg(price_close) AS avg FROM stocks
        >   WHERE exchange = 'NASDAQ' AND symbol = 'AAPL'
        >   GROUP BY year(ymd)
        >  ) s2
        > WHERE s2.avg > 50.0
```

union all

将两个或多个子查询的结果集合并到一块儿，这就要就要合并的结果集必须要有相同的列数，并且相对应的列要有相匹配的类型

```
SELECT log.ymd,log.level,log.message
  FROM(
        SELECT l1.ymd,l1.level,l1.message,'log1' AS source FROM log1 l1
        UNION ALL
        SELECT l2.ymd,l3.level,l2.message,'log2' AS source FROM log2 l2
) log
SORT BY log.ymd ASC
```

浮点比较陷阱

如果需要查询col > 0.2的结果，需要这样写：col > cast(0.2 as float)，如果只写col > 0.2，等于0.2的结果也会显示。

参考：http://blog.csdn.net/scgaliguodong123_/article/details/46944519

分析窗口函数

http://lxw1234.com/archives/2015/04/190.htm

UDF

UDF：操作单个数据行，产生单个数据行

UDAF：操作多个数据行，产生一个数据行

UDTF：操作一个数据行，产生多个数据行一个表作为输出

python udf用法

```
hive> add file path-to-py-script
hive> select transform(keyword) using 'python xxx.py' as (keyword) from tbl_name;
```

例1：udf1.py

```
import sys
for line in sys.stdin
   line = line.strip()
```

```
    print line

hive> add file udf1.py
hive> select transform(keyword) using 'python udf1.py' as keyword from tbl_name;

例2：udf2.py
import sys
for line in sys.stdin:
    frags = line.strip().split('\t')
    if len(frags) < 2:
        continue
    v1 = frags[0]
    v2 = frags[1]
    print “%s\t%s” %(v1, v2)

hive> add file udf2.py
hive> select transform(keyword1，keyword2) using 'python udf2.py' as (keyword1,
keywork2) from tbl_name;
```

使用 for line in sys.stdin 的方法要保证line中是没有\n符号的字符串，否则会出错。

java udf用法
https://cwiki.apache.org/confluence/display/Hive/HivePlugins

```
hive> add jar xx.jar;
hive> create temporary function func_name as 'com.xxx.hive.udf.xxx';
```

java udaf用法
https://cwiki.apache.org/confluence/display/Hive/GenericUDAFCaseStudy

Hive中参数的插入
在执行hive任务的时候加入参数 --hivevar即可，比如：
使用 hive --hivevar startdate=’2016-02-02’ -f connect.hive 执行connect.hive所写脚本，在脚本中使用${hivevar:startdate} 或 ${startdate}都可引用，另外需注意如果需要参数为字符串需要另外再在外面加引号 ‘${hivevar:startdate}’

其他方式还有--hiveconf VariableSubstitution

Tips

- hive默认的分隔符是’\001’，自己书写hive建表数据文件时需输入分隔符：在vim编辑模式下ctrl+v，再ctrl+a，就可以输入’\001’。
- 在Hive中创建表的时候，如果列名称和关键字一样，例如，我想创建一个表项名叫from，直接create table test(from string);是会出错的。可以使用点号（Tab键上面的·）将字段包含起来，这样即可正确创建字段。所以这里创建表的命令应该为create table test(`from` string);

by chenxiaoyu@mobvoi
E-mail: jschenxiaoyu@gmail.com
2015.11.25