JD.京东 JD.COM

大数据专家认证

HIVE简介及HQL优化V3.0

吴彪

www.jd.com

Friday, July 22, 2016

JD.京东

目录

- HIVE简介
- HIVE-MAPREDUCE
- HIVE-DDL/DML
- 数据剪裁及JOB优化
- JOIN操作优化
- 输入输出优化
- 数据去重与排序
- 数据倾斜

大数据专家认证

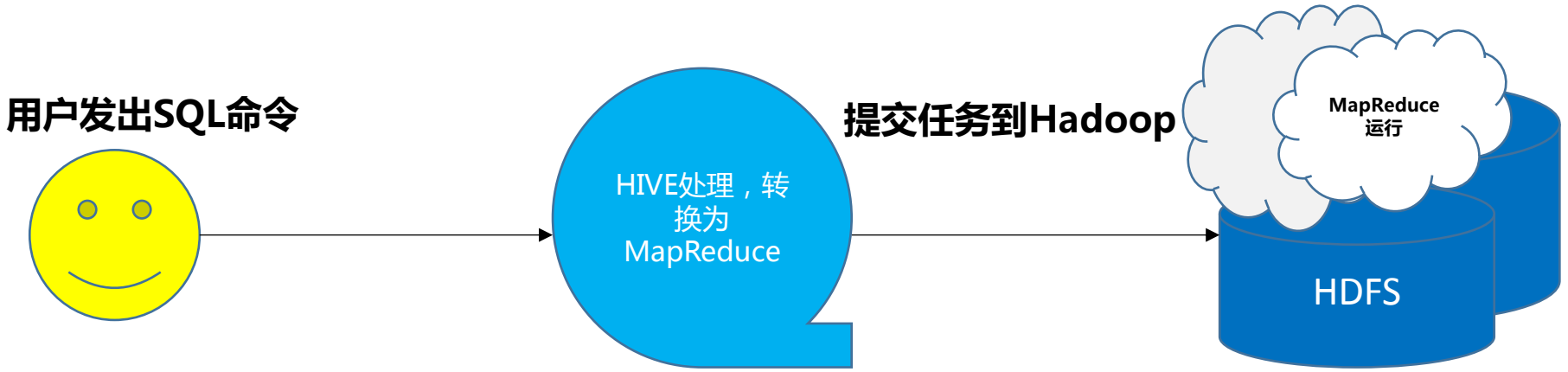
HQL优化方式及使用技巧

HIVE 简介

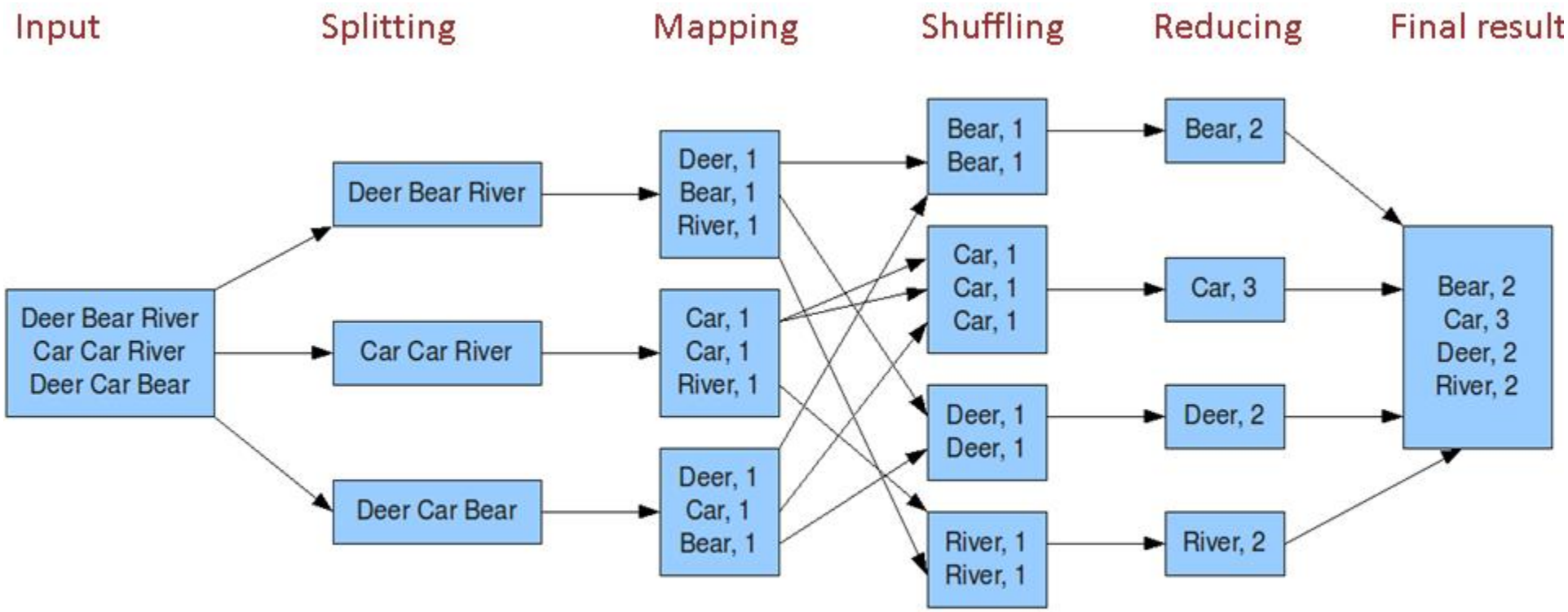
Hive是基于Hadoop的一个数据仓库工具，可以将结构化的数据文件映射为一张数据库表，并提供简单的sql查询功能，可以将sql语句转换为MapReduce任务进行运行。

Hive 定义了简单的类 SQL 查询语言，称为 HQL，它允许熟悉 SQL 的用户查询数据。同时，这个语言也允许熟悉 MapReduce 开发者的开发自定义的 mapper 和 reducer 来处理内建的 mapper 和 reducer 无法完成的复杂的分析工作。

Hive 没有专门的数据格式。



HIVE-MAPREDUCE



HIVE-DDL

➢HQL的基本数据类型及相互转换

数据类型：

基本数据类型					
类型	**描述**	**示例**	**类型**	**描述**	**示例**
TINYINT	1个字节（8位）有符号整数	1	VARCHAR	长度为1到65355	"hello"
SMALLINT	2字节（16位）有符号整数	1	CHAR	最大长度为255	"test"
INT	4字节（32位）有符号整数	1	TIMESTAMP	Integer/Floating/YYYY-MM-DD HH:MM:SS.f	2015-11-18 21:31:47
BIGINT	8字节（64位）有符号整数	1	DATE	YYYY-MM-DD	2015-11-12
FLOAT	4字节（32位）单精度浮点数	1	BINARY	二进制	byte['2','33']
DOUBLE	8字节（64位）双精度浮点数	1	arrays	ARRAY<data_type>	array('first', 'second', 'third')
BOOLEAN	true/false	TRUE	maps	MAP<primitive_type, data_type>	map('1','one','2','two','3','three')
STRING	字符串	'xia'，" xia"	structs	STRUCT<col_name : data_type>	{"col1":2,"col2":"b"}
DECIMAL	Decimal(38,18)	decimal(10,2)	union	UNIONTYPE<data_type, data_type, ...>	{3:{"a":8,"b":"eight"}}

类型转换：

cast(字段名or具体值 as 指定类型)　select cast(1 as double) from table;

HIVE-DDL

➢Hive建表（压缩表和非压缩表）

- 一个表可以拥有一个或者多个分区，每个分区以文件夹的形式单独存在表文件夹的目录下。

- 创建表，指定EXTERNAL就是外部表，没有指定就是内部表，内部表在drop的时候会从HDFS上删除数据，而外部表不会删除。

- 如果不指定数据库，hive会把表创建在default数据库下

HIVE-DDL

➢Hive建表方法（压缩表和非压缩表）

常用的创建表：

```
CREATE TABLE login(
    userid BIGINT,
    ip STRING,
    time BIGINT)
PARTITIONED BY(dt STRING)
ROW FORMAT DELIMITED
  FIELDS TERMINATED BY '\t'
STORED AS TEXTFILE;
```

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 '/user/hadoop/warehouse/page_view';
```

HIVE-DML

- **（Data Manipulation Language）数据操作语言**

1. Hive的几种JOIN方式和适用场景，Map-side join 的reduce-side join 适用场景；
2. 复杂查询的单表过滤条件使用位置，Where 和on 的区别，涉及列裁剪，分区裁剪；
3.Distribute by /sort by /cluster by /order by语法的使用和区别；
4.Hive函数使用：
sysdate()、date_format()、concat、substr、like，split等；

HIVE-DML

- **（Data Manipulation Language）数据操作语言**

5.数据倾斜的原因和解决方式；

6.Hive分区表（EDW常见的几种分区表）更有效的查询方法；

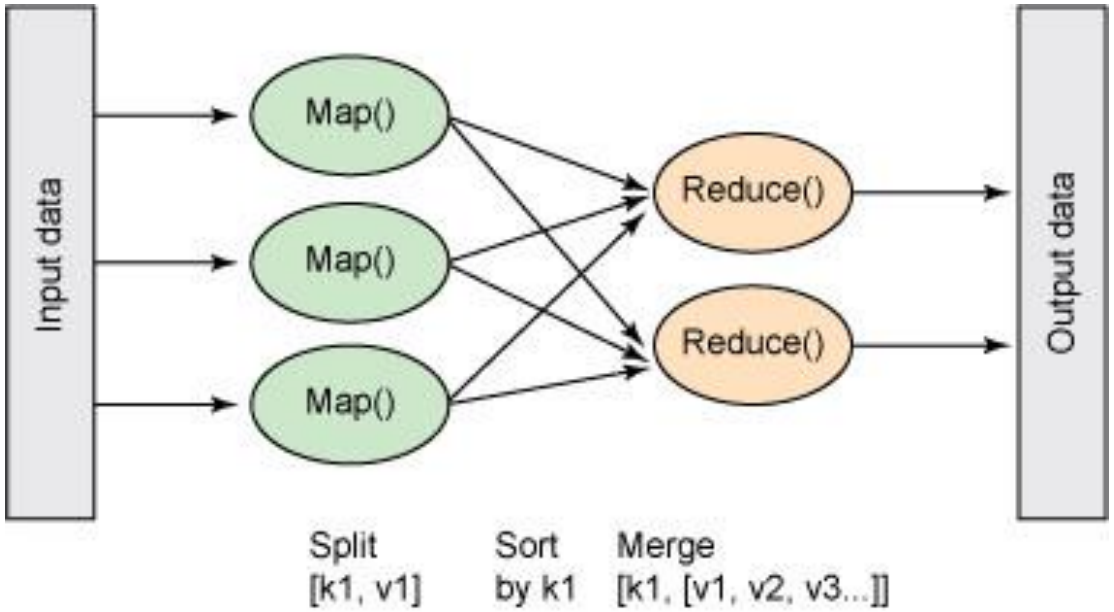
7.多表插入；

8.hive NULL 值关联；

9.动态分区；

10.数据采样；

Hive 计算原理及模型



- **优化的根本思想**

  尽早尽量过滤数据，减少每个阶段的数据量。

  减少job数。

  解决数据倾斜问题。

数据剪裁及JOB优化

- Local Model：

Select user,item from order_table limit 10;

Select * from order_table limit 10; (不会生成mapreduce程序)

- 列剪裁：

Hive 在读数据的时候，可以只读取查询中所需要用到的列，而忽略其它列

```
Select x.a,y.b from
(Select a,b From table1 Where e<10 ) x
Join
(Select a,b From table2 where e>10) y on x.a=y.a
```

在实施此项查询中，table1表有 5 列（a，b，c，d，e），Hive 只读取查询逻辑中真实需要的 3 列a、b、e，而忽略列 c，d；这样做节省了读取开销，中间表存储开销和数据整合开销。

数据剪裁及JOB优化

- 分区剪裁：

在查询的过程中减少不必要的分区

```
Select  count(orderid) From order_table  where
to_date(sale_time)=  '2014-03-03'  and
hour(to_date(sale_time))=10


Select  count(orderid) From order_table   where
dt =  '2014-03-03'  and  to_date(sale_time)=  '2014-03-03'
and hour(to_date(sale_time))=10
```

数据剪裁及JOB优化

➢分区剪裁：

Explain dependency 语法，获取 input table和input partition

```
hive> Explain dependency   select count(1) from
            app_cmo_supp_sku_vlt where dt>='2014-06-01';
hive>
{"input_partitions":[{"partitionName":"app@app_cmo_supp
_sku_vlt@dt=2014-06-
03"},{"partitionName":"app@app_cmo_supp_sku_vlt@dt=2
014-06-
04"},{"partitionName":"app@app_cmo_supp_sku_vlt@dt=2
014-06-
05"}],"input_tables":[{"tablename":"app@app_cmo_supp_sk
u_vlt","tabletype":"EXTERNAL_TABLE"}]}
```

数据剪裁及JOB优化

➢利用hive 的优化机制减少JOB数：

不论是外关联 outer join还是内关联 inner join， 如果Join的 key相同，不管有多少个表，都会合并为一个MapReduce 任务。

SELECT a.val, b.val, c.val FROM a JOIN b ON (a.key = b.key1) JOIN c ON (c.key= b.key1) -> **1个JOB**

SELECT a.val, b.val, c.val FROM a JOIN b ON (a.key = b.key1) JOIN c ON (c.key = b.key2) -> **2个JOB**

数据剪裁及JOB优化

➢JOB输入输出优化：

善用muti-insert、union all，不同表的union all相当于multiple inputs,同一个表的union all,相当map一次输出多条

```
1、insert overwrite table tmp1
        select ...    from a    where 条件1;
    insert overwrite table tmp2
        select ...    from a  where 条件2;

2、from a
        insert overwrite table tmp1
            select ...   where 条件1
        insert overwrite table tmp2
            select ...    where 条件2;
```

JOIN操作及优化

➢避免笛卡尔积：

```
SELECT   ....
from   woa_all_device_info_his A
left outer join(
        select *
        from   woa_all_info_his B
        where  (B.mobile <> 'unknown' or B.imsi <> 'unknown')
        and    B.imei <> 'unknown'
        and    B.pt = '$data_desc'
       ) C
on     A.app_id = C.app_id and A.imei = C.imei
```

JOIN操作及优化

➢数据过滤：

在JOIN前过滤掉不需要的数据

1、SELECT a.val, b.val FROM a LEFT OUTER JOIN b ON (a.key=b.key)
WHERE a.ds='2009-07-07' AND b.ds='2009-07-07'

2、SELECT x.val, y.val FROM
(select key,val from a where a.ds= ‘2009-07-07’) x
LEFT OUTER JOIN
(select key,val from b where b.ds= ‘2009-07-07’) y
ON x.key=y.key

JOIN操作及优化

- 小表放前大表放后原则：

在编写带有join操作的代码语句时，应该将条目少的表/子查询放在Join操作符的左边。
因为在 Reduce 阶段，位于 Join 操作符左边的表的内容会被加载进内存，载入条目较少的表
可以有效减少 OOM（out of memory）即内存溢出。所以对于同一个 key来说，对应的 value
值小的放前，大的放后

JOIN操作及优化

- Mapjoin()：

当小表与大表JOIN时，采用mapjoin ,即在map端完成。同时也可以避免小表与大表JOIN 产生的数据倾斜。

```
SELECT /*+ MAPJOIN(b) */ a.key, a.value
          FROM a join b on a.key = b.key
```

JOIN操作及优化

➢LEFT SEMI JOIN ：

LEFT SEMI JOIN 是 IN/EXISTS 子查询的一种更高效的实现，0.13版本以前不支持 IN/EXISTS

1、SELECT a.key,a.value FROM a WHERE a.key in (SELECT b.key FROM b) （hive-0.13之前不支持）

2、通过left outer join 实现 in 查询
select a.key,a.value from a left outer join b on a.key=b.key where b.key is not null

3、通过left semi join 实现 in
SELECT a.key ,a.value FROM a LEFT SEMI JOIN b on (a.key = b.key)

4、通过 join 实现 in
SELECT a.key,a.value FROM a JOIN b on (a.key = b.key)

JOIN操作及优化

- LEFT SEMI JOIN ：

限制条件:

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

Left semi join 与 JOIN 的区别:

B表有重复值的情况下 left semi join 产生一条，join 会产生多条

JOIN操作及优化

➢LEFT SEMI JOIN 与 JOIN 区别 ：

A 表	
a	1
b	2
c	3
d	4
e	5
f	6
g	7

B表	
a	name1
a	name1
b	name2
c	name3
d	name4

JOIN

a	1	name1
a	1	name1
b	2	name2
c	3	name3
d	4	name4

LEFT SEMI JOIN

a	1	name1
b	2	name2
c	3	name3
d	4	name4

输入输出优化

➢合理使用动态分区：

- SET hive.exec.dynamic.partition=true;
- SET hive.exec.dynamic.partition.mode=nonstrict;

- ```
  create table lxw_test1 (
  sndaid string,
  mobile string
  ) partitioned by (dt  STRING)
  stored as rcfile;

  insert overwrite table lxw_test1 partition (dt)
  select sndaid,mobile,dt from woa_user_info_mes_tmp1
  ```
  ```

输入输出优化

➢union all 优化 ：

利用hive 对UNION ALL的优化的特性(0.13版本可以直接union)
hive对union all优化只局限于非嵌套查询

```
1、select * from
        (select  ci,c2,c3 from t1 Group by c1,c2,c3
        Union all
        Select  c1,c2,c3 from t2  Group by c1,c2,c3
    ) t3;

2、select * from
        (
        select * from t1
        Union all
        Select * from t2
        ) t3
    Group by c1,c2,c3;
```

JOB: 3

JOB: 1

输入输出优化

➢合理使用 union all ：

不同表太多的union ALL,不推荐使用;

通常采用建临时分区表，将不同表的结果insert到不同的分区(可并行)，最后再统一处理;

```
INSERT overwrite TABLE lxw_test (flag = '1')
SELECT sndaid,mobile FROM lxw_test1;

INSERT overwrite TABLE lxw_test (flag = '2')
SELECT sndaid,mobile FROM lxw_test2;

INSERT overwrite TABLE lxw_test (flag = '3')
SELECT sndaid,mobile FROM lxw_test3;
```

输入输出优化

- 合理使用 UDTF：

```
select col1,col2,newCol from myTable LATERAL VIEW
explode(myCol) adTable AS newCol
```

说明：

执行过程相当于单独执行了两次读取，然后union到一个表里，但JOB数只有一个。

同样myCol 也需要为数组类型，但日常中我们多数情况下是string 类型经过split 函数拆分后获取数组类型 。

输入输出优化

➢合理使用 UDTF：

id	user	name	all_citys
id1	user1	name1,	北京,天津
id2	user2	name2,	上海,河北

select id ,user,name,city from myTable LATERAL VIEW explode(split(all_citys,',')) adTable AS city

id	user	name	city
id1	user1	name1	北京
id1	user1	name1	天津
id2	user2	name2	上海
id2	user2	name2	河北

输入输出优化

➢多粒度计算优化：

应用UDTF 优化：按不同维度进行订单汇总。

Order_id	Province	City	County	Sales

```
1、
select * from (
      select   '1',province,sum(sales) from order_table group by province
      union all
      select   '2',city,sum(sales) from order_table group by city
      Union all
      select   '3',county,sum(sales) from order_table group by county
) df
```

3次读取order_table,4个JOB

输入输出优化

➢多粒度计算优化：

应用UDTF 优化：按不同维度进行订单汇总。

```
2、
select type,code,sum(sales) from (
      select split(part,'_')[1] as type,
               split(part,'_')[0] as code ,
              sales from order_table LATERAL VIEW
      explode(split(concat(province,'_1-',city,'_2-',county,'_3'),'-'))
      adTable AS part
) df group by type,code
```

数据去重与排序

➢DISTINCT 与 GROUP BY ：

尽量避免使用 DISTINCT 进行排重，特别是大表操作，用 GROUP BY 代替。

```
1、Select distinct  key from a

2、Select key from a  group by key
```

数据去重与排序

➢排序优化：

只有order by 产生的结果是全局有序的，可以根据实际场景进行选择排序。

1、Order by 实现全局排序，一个reduce实现，由于不能并发执行，所以效率偏低

2、Sort by 实现部分有序，单个reduce输出的结果是有序的，效率高，通常和DISTRIBUTE BY关键字一起使用（ DISTRIBUTE BY关键字 可以指定map 到 reduce端的分发key ）

3、CLUSTER BY col1 等价于DISTRIBUTE BY col1 SORT BY col1 但不能指定排序规则

数据倾斜

➢数据倾斜：

任务进度长时间维持在99%（或100%），查看任务监控页面，发现只有少量（1个或几个）reduce子任务未完成。因为其处理的数据量和其他reduce差异过大。
单一reduce的记录数与平均记录数差异过大，通常可能达到3倍甚至更多。 最长时长远大于平均时长。

HQL案例实战

➢HIVE –实例

- 统计10月份各省份下单数TOP100的用户信息及订单数量

用户表：

ID	NAME	PROVINCECODE	CITYCODE	AGE
1	张三	P00001	C00001	26

城市表：

CODE	NAME
P00001	北京

订单表：

ID	USERID	ITEMID	CREATEDATE
234	1	100423	2014-10-11 15:20:20

HQL案例实战

➢HIVE –实例（旧版本row_number用法,不建议使用）

```
select /*+ MAPJOIN(t2)*/ t1.id,t1.name,t2.cityname,t1.nums from
  select id,name,provinceid,nums from (
    select id,name,provinceid,nums ,row_number(provinceid) rank from (
        select  id,name,provinceid,nums from        (
            select a.id,a.name,a.provinceid,count(1) as nums   from  (
                select id,name,province from user_da where dt='2014-11-11')   a
                join (select id,userid,itemid from orders_chain where dp in  ( 'ACTIVE','HISTORY')
                and to_date(createdate)>='2014-10-01' and to_date(createdate)<='2014-10-31' )
                  b
                    on a.id=b.user   group by a.id,a.name,a.provinceid
          ) x  distribute by  provinceid sort by provinceid,nums desc
    )  y
  ) z where rank <=100
) t1  left outer join dim_citys t2 on t1.provinceid=t2 .citycode
```

HQL案例实战

➢HIVE –实例（新row_number用法）

```
CREATE temporary function row_number  as 'org.apache.hadoop.hive.ql.udf.generic.GenericUDAFRowNumber';
select /*+ MAPJOIN(t2)*/ t1.id,t1.name,t2.cityname,t1.nums from
  select id,name,provinceid,nums from (
    select id,name,provinceid,nums ,row_number() rank over (partition by provinceid order by nums desc) rank
from (
            select a.id,a.name,a.provinceid,count(1) as nums   from  (
                select id,name,province from user_da where dt='2014-11-11')  a
                join (select id,userid,itemid from orders_chain where dp in  ( 'ACTIVE','HISTORY')
                and to_date(createdate)>='2014-10-01' and to_date(createdate)<='2014-10-31' )       b
                    on a.id=b.user   group by a.id,a.name,a.provinceid
    )  y
  ) z where rank <=100

) t1  left outer join dim_citys t2 on t1.provinceid=t2 .citycode
```

使用技巧

➢有用的新特性：

1、指定列之间的分隔符可以用下面的命令实现：

```
hive> insert overwrite local directory /home/dd_edw/documents/result'
hive> row format delimited
hive> fields terminated by '\t'
hive> select * from test;
```

2、group by 语法增强，group by除了可以跟column alias，也可以跟column position

比如：

```
select f1(col1), f2(col2), f3(col3), count(1) \
group by f1(col1), f2(col2), f3(col3);
```

可以写成

```
select f1(col1), f2(col2), f3(col3), count(1) group by 1, 2, 3;
```

使用技巧

➢有用的新特性：

3、ALTER VIEW view_name AS select_statement

4、 SHOW CREATE TABLE ([db_name.]table_name|view_name)

5、Explain dependency 语法，获取 input table和input partition

6、实现了TRUNCATE，可以删除HDFS上面相关表格存储的数据，但是会保持表和metadata的完整性。

使用技巧

➢新版本注意事项：

1、新版本字段别名

以字母，数字，下划线组合 。
但不能以数字开头 。

2、涉及精度较高运算时，精度会与旧版本有微小差异

新版本对执行计划有调优，所以各别计算顺序，精度取舍与旧版会有不同，造成结果的微小差异。

大数据专家认证

谢谢！
Thank you!

京东 . 大数据平台

北京市朝阳区北辰西路8号北辰世纪中心A座10层
10F Building A, North-Star Century Center, 8 Beichen West Street,
Chaoyang District, Beijing 100101