**Hive SQL常用设置**

[Yarn 内存分配管理机制及相关参数配置](https://blog.csdn.net/suifeng3051/article/details/48135521)

1. **加大节点内存，解决查询数据量大导致的任务失败**

* set mapreduce.map.memory.mb=8192;
* set mapreduce.reduce.memory.mb=8192;
* set mapreduce.map.java.opts=-Xmx6144m;
* set mapreduce.reduce.java.opts=-Xmx6144m;

1. **设置写文件的大小，尽量接近于hdfs的块大小256M**

* hive表如果是数据量巨大的小文件，那么查询和spark读取速度将非常慢
* set hive.merge.mapfiles = true ;
* set hive.merge.mapredfiles = true;
* set hive.merge.size.per.task = 100000000;
* set hive.merge.smallfiles.avgsize = 100000000;

1. **设置写表文件压缩**

* set hive.exec.compress.output = true; -- 压缩文件，默认false，是否对输出结果压缩
* set mapred.output.compress = true;
* set mapred.output.compression.codec = org.apache.hadoop.io.compress.GzipCodec;
* set io.compression.codecs = org.apache.hadoop.io.compress.GzipCodec;

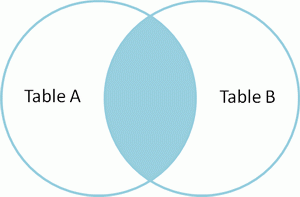
1. **设置动态写分区配置**

* 最多能写100个分区，不宜超过这个数
* set hive.exec.dynamic.partition = true;
* set hive.exec.dynamic.partition.mode = nonstrict;
* <http://dp.pt.xiaomi.com/job/job_details/281474>

1. **设置子作业并行执行**

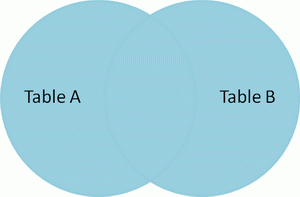
* 如果作业特别长包含多个join或union，可以启用此设置
* set hive.exec.parallel=true;
* set hive.exec.parallel.thread.number=8; //设置并行数
* <http://dp.pt.xiaomi.com/task/21609416>

1. **Join操作**
2. **A∩B：inner join(join) hive里join默认为inner join**



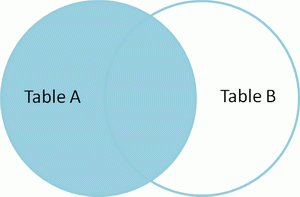
* id name id name
* -- ---- -- ----
* 1 Pirate 1 Rutabaga
* 2 Monkey 2 Pirate
* 3 Ninja 3 Darth Vader
* 4 Spaghetti 4 Ninja
* select \* from TableA inner join TableB on TableA.name = TableB.name
* id name id name
* -- ---- -- ----
* 1 Pirate 2 Pirate
* 3 Ninja 4 Ninja

1. **A∪B：full outer join**



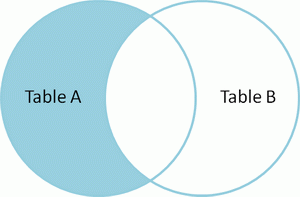
* select \* from TableA full outer join TableB on TableA.name = TableB.name
* id name id name
* -- ---- -- ----
* 1 Pirate 2 Pirate
* 2 Monkey null null
* 3 Ninja 4 Ninja
* 4 Spaghetti null null
* null null 1 Rutabaga
* null null 3 Darth Vader

1. **A：left outer join**



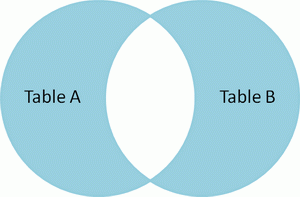
* select \* from TableA left outer join TableB ON TableA.name = TableB.name
* id name id name
* -- ---- -- ----
* 1 Pirate 2 Pirate
* 2 Monkey null null
* 3 Ninja 4 Ninja
* 4 Spaghetti null null

1. **A-B**



* select \* from TableA left outer join TableB ON TableA.name = TableB.name where TableB.id IS null
* id name id name
* -- ---- -- ----
* 2 Monkey null null
* 4 Spaghetti null null

1. **|A-B|**



* select \* from TableA full outer join TableB on TableA.name = TableB.name where TableA.id is null or TableB.id is null
* id name id name
* -- ---- -- ----
* 2 Monkey null null
* 4 Spaghetti null null
* null null 1 Rutabaga
* null null 3 Darth Vader

1. **Join操作之后进行group by去重**

* group by可以替代distinct的操作；也可以goup by所有属性进行重复行除重
* 在join操作后可能会出现重复行，是因为进行join操作时某个表有重复数据导致的，这时可以进行一下group by操作去除重复数据

1. **时间格式转换(Time format)**
2. **日期转时间戳**

* 获取当前时间戳：unix\_timestamp()，返回类型：int
* 到期日期大于今天：unix\_timestamp(due\_time, "yyyy-MM-dd HH:mm:ss") > unix\_timestamp("${date}", "yyyyMMdd")
* unix\_timestamp('2018-09-17 11:49:43')=1537156183

1. **时间戳转日期**

* 时间戳转为时间格式：from\_unixtime(unix\_timestamp(),'yyyy-MM-dd')，返回类型：string，函数第二个参数定义返回类型，例子中的‘yyyy-MM-dd’，也可为‘yyyyMMdd’
* from\_unixtime(1537156183) = 2018-09-17 11:49:43
* cast (from\_unixtime(unix\_timestamp('2018-09-17 11:49:43'), 'yyyyMMdd') as int)=20180917

1. **时间差计算**

* datediff('2018-09-17','2018-09-16')=1
* datediff(from\_unixtime(unix\_timestamp(x\_time), "yyyy-MM-dd"), from\_unixtime(unix\_timestamp(), "yyyy-MM-dd"))算的是x\_time减去当前时间的天数

1. **算日期是周几**

* pmod(datediff(X, '1920-01-01')-3,7), X='yyyy-MM-dd'
* pmod(datediff('2019-12-22','1920-01-01')-3,7)=0 （周日到周六对应0到6）
* 如果是yyyyMMdd类型的数字，就是date类型，可以这样：pmod(datediff(concat(substr(${date},1,4),'-',substr(${date},5,2),'-',substr(${date},7,2)), '1920-01-01')-3,7)

1. **unix\_timestamp()与${date}作为作业里的静态参数和动态参数**

* datediff(concat(substr(${date},1,4),'-',substr(${date},5,2),'-',substr(${date},7,2)),from\_unixtime(unix\_timestamp(), "yyyy-MM-dd"))
* 如果某时刻,比如2018-10-19 17:41:39，定时任务需要执行过去一个月的任务，这三十个任务里的unix\_timestamp()值都是一样的，是当下的时间戳值15399420999(2018-10-19 17:41:39)，${date}在每个任务里值不一样，datediff(concat(substr(${date},1,4),'-',substr(${date},5,2),'-',substr(${date},7,2)),from\_unixtime(unix\_timestamp(), "yyyy-MM-dd"))的值就是${date}与20181019的天数差-30,-29,.......,-1。这样就可以通过datediff在hive语句里设置参数。仅举一例，不一而足。参见http://dp.pt.xiaomi.com/job/job\_details/160746

1. **lateral view explode一行变多行,一行变指定多行**
2. **lateral view explode一行变多行**

* <http://dp.pt.xiaomi.com/job/job_details/121158>

1. **一行变指定多行，实现类似range(N)功能**

* [hive实现range(n)一行变指定多行](http://dp.pt.xiaomi.com/job/job_details/286462)

1. **row\_number() over(partition by A, B, C order by D)**

* 分组内排序，二次row\_number查找分组内TopN
* <http://dp.pt.xiaomi.com/job/job_details/121158>

1. **读json数组**

* 读json array例子：http://dp.pt.xiaomi.com/job/job\_details/271954
* json里面如果有两层或以上的数组嵌套，SQL就无能为力，所以打点设计不要在json里放两层或以上的数组嵌套

1. **with q1 as (Query)**

* 数据加到内存提高查询速度，适用于对一张表进行多次条件过滤或不同维度group by的查询
* <http://dp.pt.xiaomi.com/job/job_details/174448>

1. **with cube**

* group by A,B, C with cube/with rollup/grouping sets(A,B,C,(A,B),...,(A,B,C)) 生成多维度统计数据
* 举个例子，假设想从mitv\_log表里统计A,B,C所有维度(A,B,C,AB,AC,BC,ABC)的vv与uv，可用with cube: select A,B,C, count(\*) as vv, count(disticnt concat(deviceid,ptf)) as uv from mitv\_log where date=${date-1} group by A,B,C with cube
* 安装失败的定时作业：http://dp.pt.xiaomi.com/job/job\_details/237768，这个例子里统计了app\_version, ptf, package, isSysApp, silent, errorType, error七个属性参与的任意维度总共128类维度的安装失败数，尝试安装数和含外因的安装数
* 指定某些维度可用grouping sets(A,B,C)替代with cube，grouping sets(A,B,C)只统计分A,B,C三种维度的数据
* with rollup实用的少，详细可百度
* 在求uv，也就是有count(distinct) 类的去重统计时，with cube默认参与统计的维度不超过5个，对应的参数**hive.new.job.grouping.set.cardinality**默认值是32。当维度查过5个时例如有6个维度参与统计: group by A,B,C,D,E,F with cube，此时需要设置参数set hive.new.job.grouping.set.cardinality=64; **只求vv的话没有此类限制**。

1. **去除空格，NULL和空**

* '[\\s]+|[\\u3000]+|[\,]'可以匹配字符串中的空格：regexp\_replace(' 66666 ', '[\\s]+|[\\u3000]+|[\,]','1')=1666661
* nvl()可以将NULL转为其他值：假设x=NULL，则nvl(x,'未知')='未知'
* if(x='',x,y)去除空值：如果某个属性值是空值，即x=''，它不是NULL，值里也没有空格，替换的时候可用if语句：if(x='','未知',x)='未知'
* [去除空字符和NULL](http://dp.pt.xiaomi.com/job/job_details/288155)

1. **Hive SQL数据倾斜解决办法**
2. **正常情况的sum count max min 不会出现数据倾斜问题**
3. **出现数据倾斜的情况**
4. **如果单个key里出现几百万或更多的数据（比如一些热门节目的播放用户）做Join操作**
5. **对大表查询的count(distinct XXX)作业，比如使用with cube计算各维度的uv**
6. **优化建议：尽量使用最简单粗暴的方法解决，时效最高。下面从最基础起步，不止针对出现数据倾斜的作业，哪条合适选哪条**
7. **减少查询的列/分区/去除非必要异常值(空值/NULL/-1等等)**

* select a,b from table 而不是 select \* from table
* select \* from (select a,b from table where c = 123)t 而不是 select \* from (select a,b from table)t where t.c = 123；

1. **小表加入内存，使用广播(适当使用MapJoin/With)**

* 小表join大表操作时 将小表放在join左边

1. **join倾斜优化**

* set hive.skewjoin.key=100000; //join的键对应的记录条数超过这个值则会进行分拆,值根据具体数据量设置
* set hive.optimize.skewjoin=true; //如果join 过程出现倾斜应该设置为true

1. **group by倾斜优化**

* set hive.map.aggr=true;
* set hive.groupby.mapaggr.checkinterval = 100000 (默认);
* set hive.map.aggr.hash.min.reduction=0.5(默认);//两个参数的意思是：预先取100000条数据聚合,如果聚合后的条数/100000>0.5，则不再聚合
* set hive.groupby.skewindata=true；//决定 group by操作是否支持倾斜的数据

1. **disticnt用group by和count代替**

* 一般的count(distinct)数据工厂性能都OK，无需写更繁琐的嵌套SQL绕路，如果99%几十分钟或几个小时，看看group by再count是否可行，这样的话，vv和uv就得分开算
* <http://dp.pt.xiaomi.com/job/job_details/245110>

1. **1到5都不能解决问题，可以考虑热点key枚举一一计算，再和剩下的union**
2. **加一列随机数，多做一层查询，真需要到这一步，考虑下表设计是否有问题**
3. **真正意义的数据倾斜解决不了，尽量避免，尽量选择上面最靠前的解决办法**
4. **抽样数据(limit限制调整)**
5. 一般情况下，Limit语句还是需要执行整个查询语句，然后再返回部分结果。
6. 有一个配置属性可以开启，避免这种情况---对数据源进行抽样
7. hive.limit.optimize.enable=true --- 开启对数据源进行采样的功能
8. hive.limit.row.max.size --- 设置最小的采样容量
9. hive.limit.optimize.limit.file --- 设置最大的采样样本数
10. 缺点：有可能部分数据永远不会被处理到