Hive10 模式设计

> Hive实现和使用的方式和传统的 关系型数据库是非常不同的。

按天划分的表

按天划分的表就是一种模式，其通常会在表名中加入一个时间戳。例如：supply_2011_01_01、supply_2011_01_02。这种每天一张表的方式在数据库领域是一种反模式的一种方式，但是因为实际情况下数据集增长得很快，这种方式应用还是比较广泛的。

```
hive> CREATE TABLE supply_2011_01_02(id int, part string, quantity int);
hive> CREATE TABLE supply_2011_01_03(id int, part string, quantity int);
hive> CREATE TABLE supply_2011_01_04(id int, part string, quantity int);

hive> ... load data ..

hive> SELECT part,quantity supply_2011_01_02
    > UNION ALL
    > SELECT part, quantity from supply_2011_01_03
    > WHERE quantity < 4;
```

对于Hive,这种情况下应该使用分区表，Hive通过WHERE子句中的表达式来选择查询所需要的指定分区。

```
hive> CREATE TABLE supply (id int, part string, quantity int)
    > PARTITIONED BY (int day);

hive> ALTER TABLE supply add PARTITION (day=20110102);
hive> ALTER TABLE supply add PARTITION (day=20110103);
hive> ALTER TABLE supply add PARTITION (day=20110104);

hive> ... load data...

hive> SELECT part,quantity FROM supply
    > WHERE day>=20110102 AND day<20110103 AND quantity < 4;
```

关于分区

> Hive 中的分区的功能是非常有用的，这是因为Hive通常要对输入进行全盘扫描，来满足查询条件。通过创建很多分区确实可以优化一些查询，但是同时可能会对其他一些重要的查询不利

```
hive> CREATE TABLE weblogs(url string, time long)
    > PARTITIONED BY (day int, state string, city string);

hive> SELECT * FROM weblogs WHERE day=20110102;
```

> MapReduce 会将一个任务(job) 转换成多个任务（task)，默认情况下，每个task都是一个新的JVM实例，都需要开启和销毁的开销。对于小文件，每个文件都会对应一个task。在一些情况下，JVM开启和销毁的时间中销毁可能会比实际处理数据的时间消耗更长。

- 按时间分区

按照时间范围进行分区的一个好的策略就是按照不同的时间粒度来确定合适大小的数据累积量，而且安装这个时间粒度。随着时间推移，分区数量的增长是均匀的，而且每个分区下包含的文件大小至少是文件系统中块的大小或者数倍。这个平衡可以保持分区足够大，从而一般情况下查询的数据吞吐量。

```
hive> CREATE TABLE weblogs (url string, time long, state string, city string)
    > PARTITIONED BY (day int);
hive> SELECT * FROM weblogs WHERE day=20110102;
```

- 使用两个级别的分区，并且使用不同维度

```
hive> CREATE TABLE weblogs (url string, time long, city string)
    > PARTITIONED BY (day int, state string);
hive> SELECT * FROM weblogs WHERE day = 20110102
```

因为有一些州可能会比其他州具有更多的数据，用户可能会发现map task 处理数据时出现不均。

唯一键和标准化

避免标准化

> 避免标准化的主要原因是为了最小化磁盘寻道，比如那些通常需要外键关系的情况，非标准化数据允许被扫描或写入到大的、连续的磁盘存储区域，从而优化磁盘驱动器的I/O性能。然而，非标准化数据可能导致数据重复，而且有更大的导致数据不一致的风险。

```
CREATE TABLE employees(
    name            STRING,
    salary          FLOAT,
    subordinates    ARRAY<STRING>,
    deductions      MAP<STRING, FLOAT>,
    adress          STRUCT<street:STRING, city:STRING,  state:STRING, zip:INT>
    )
```

打破传统的设计原则：

- 非正式地使用name作为主键，名字并不是唯一地。
- 使用subordinates数组字段保存了员工所有下属地一个外键
- 使用map来记录各项收税和扣税

同一份数据多种处理

> 可以从一个数据源产生多个数据聚合，而无需每次聚合都要重新扫描一次。对于大地数据输入记来说，这个优化可以节约非常可观地时间。

- 2个查询会从源表history表读取数据，然后导入到2个不用的表中：

```
hive> INSERT OVERWRITE TABLE sales
    > SELECT * FROM history WHERE action='purchased';
hive> INSERT OVERWRITE TABLE credits
    > SELECT * FROM history WHERE action='returned';
```

上面查询，效率低下，语法正确，下面语句只需要查询一次表即可：

```
hive> FROM history
    > INSERT OVERWRITE sales SELECT * WHERE action='purchased'
    > INSERT overwrite credits SELECT * WHERE action='returned';
```

对于每个表的分区

> 由于查询或者原始数据处理的某个步骤出现问题而导致需要对好几天的输入数据重跑ETL过程，者手用户可能就需要执行那些一天执行一次的处理过程，来保证所有任务完成之前不会有job将临时表覆盖重写。

```
$ hive -hiveconf dt=2011-01-01
hive> INSERT OVERWRITE table distinct_ip_in_logs
    > PARTITION(hit_date=${dt})
    > SELECT distinct(ip) as ip from weblogs
    > WHERE hit_date ='${hiveconf:dt}';

hive> CREATE TABLE state_city_for_day (state string, city string)
    > PARTITIONED BY (hit_date string)

hive> INSERT OVERWRITE table state_city_for_day PARTITION(${HIVECONF:DF})
    > SELECT distinct(state, city) FROM distinct_ip_in_logs
    > JOIN geodate ON (distinct_ip_in_logs.ip = geodate.ip)
    > WHERE (hit_date='${hiveconf:dt}');
```

分桶表数据存储

> 分桶是将数据集分解成更容易管理的若干部分的另一个技术

例如：假设有个表的一级分区是dt，代表日期；二级分区是user_id，那么这种划分方式可能会导致太多的小分区。如果对weblog进行分桶，并使用user_id字段作为分桶字段，则字段会根据用户指定的值进行哈希分发到桶中。同一个user_id下的记录通常会存储到同一个桶内。

```
hive> CREATE TABLE weblog (user_id INT, url STRING, source_ip STRING)
    > PARTITIONED BY (dt STRING)
    > CLUSTERED BY (user_id) INTO 96 BUCKETS;
```

不过，将数据正确地插入到表地过程完全取决于用户自己，CREATE TABLE语句中所规定地信息仅仅定义了元数据，而不影响实际填充表地命令。

```
hive> SET hive.enforce.bucketing = true;

hive> FROM raw_logs
    > INSERT OVERWRITE TABLE weblog
    > PARTITION (dt = '2019-02025')
    > SELECT user_id, url source_ip WHERE dt='2019-02-25';
```

- bucketing参数：如果没有使用这个属性，那么我们就需要自己设置和分桶个数相匹配地reducer个数。
- 例如： 使用 set mapred.reduce.task=96，然后在INSERT语句中，需要SELECT 语句后增加CLUSTER BY语句。

> 分桶的优点：
>
> - 因为桶的个数是固定的，所以没有数据波动。桶非常适合抽样。
> - 有利于执行高效的map-side JOIN

为表增加列

> Hive允许在原始数据文件之上定义一个模式，而不像很多的数据库那样，必须以特定的格式转换和导入数据。

SerDe抽象

一个SerDe通常是从左到右进行解析的，通过指定的分隔符将行分解成列。SerDe通常是非常宽松的。如果某行的字段个数比预期的要少，那么缺少的字段将返回null，如果某行的字段个数比预期的要多，那么多出的字段会被省略掉。

```
hive> CREATE TABLE weblogs(version LONG, url STRING)
    > PARTITIONED BY (hit_date int)
    > ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t';

hive> ! cat log1.txt
1 /mystuff 20110101
1 /toys 20110101

hive> LOAD DATA LOCAL INPATH 'log1.txt' int weblogs partition(20110101);

hive> SELECT * FROM weblogs;
1  /mystuff 20110101
1  /toys    20110101
```

随着时间的推移，可能会为底层数据增加一个新字段。

```
hive> ! cat log2.txt
2 /cars bob
2 /stuff terry

hive> ALTER TABLE weblogs ADD COLUMNS (user_id string)

hive> LOAD DATA LOCAL INPATH 'log2.txt' int weblogs partition(20110102)

hive SELECT * FROM weblogs
1 /mystuff  20110101 NULL
1 /toys     20110101 NULL
2 /cars     20110102 bob
2 /stuff    20110102 terry
```

使用列存储表

> Hive提供了一个列式SerDe来以混合列格式存储信息，虽然这个格式是可以用于任意类型的数据的，不过对于某些数据集使用这种方式是最优的

重复数据

假设有足够多的行，像state字段和age字段这样的列将会有很多重复的数据。

state	uid	age
NY	Bob	40
NJ	Sara	32
NY	Peter	14
NY	Sandra	4

多列

非常多的字段

state	uid	age	server	tz	many_more..
NY	Bob	40	web1	est	stuff
NJ	Sara	32	web1	est	stuff
NY	Peter	14	web3	pst	stuff
NY	Sandra	4	web45	pst	stuff

查询通常只会用到一个字段或者很少的一组字段，基于列式存储会使分析表数据行驶执行得更快：

```
hive> SELECT distinct(state) FROM weblogs;
NY
NJ
```