文件存储格式

Hive支持的存储数据的格式主要有：TEXTFILE 、SEQUENCEFILE、ORC、PARQUET。

TEXTFILE是默认的格式，ORC、PARQUET是列存储格式，占用空间和查询效率是不同的

使用 建表时候 stored as +格式类型

一：建表语句差别

```
create table if not exists text(
a bigint
) partitioned by (dt string)
row format delimited fields terminated by '\001'
location '/hdfs/text/';

create table if not exists orc(
a bigint)
partitioned by (dt string)
row format delimited fields terminated by '\001'
stored as orc
location '/hdfs/orc/';

create table if not exists parquet(
a bigint)
partitioned by (dt string)
row format delimited fields terminated by '\001'
stored as parquet
location '/hdfs/parquet/';
```

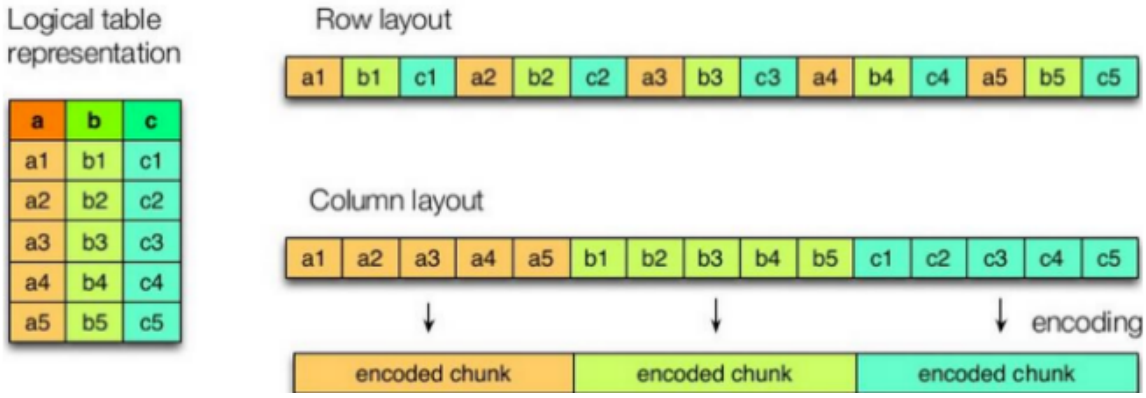
二：HDFS存储对比

parquet	orc	text
709M	275M	1G
687M	249M	1G
647M	265M	1G

三：查询时间对比

parquet	orc	text
36.451	26.133	42.574
38.425	29.353	41.673
36.647	27.825	43.938

四:列式存储和行式存储



1．行存储的特点

查询满足条件的一整行数据的时候，列存储则需要去每个聚集的字段找到对应的每个列的值，行存储只需要找到其中一个值，其余的值都在相邻地方，所以此时行存储查询的速度更快。

2．列存储的特点

因为每个字段的数据聚集存储，在查询只需要少数几个字段的时候，能大大减少读取的数据量；每个字段的数据类型一定是相同的，列式存储可以针对性的设计更好的设计压缩算法。

TEXTFILE和SEQUENCEFILE的存储格式都是基于行存储的；

ORC和PARQUET是基于列式存储的。

4.2 TextFile格式

默认格式，数据不做压缩，磁盘开销大，数据解析开销大。可结合Gzip、Bzip2使用，但使用Gzip这种方式，hive不会对数据进行切分，从而无法对数据进行并行操作。

4.3 Orc格式

Orc (Optimized Row Columnar)是Hive 0.11版里引入的新的存储格式。

如图6-11所示可以看到每个Orc文件由1个或多个stripe组成，每个stripe一般为HDFS的块大小，每一个stripe包含多条记录，这些记录按照列进行独立存储，对应到Parquet中的row group的概念。每个Stripe里有三部分组成，分别是Index Data，Row Data，Stripe Footer：

1）Index Data：一个轻量级的index，默认是每隔1W行做一个索引。这里做的索引应该只是记录某行的各字段在Row Data中的offset。

2）Row Data：存的是具体的数据，先取部分行，然后对这些行按列进行存储。对每个列进行了编码，分成多个Stream来存储。

3）Stripe Footer：存的是各个Stream的类型，长度等信息。

每个文件有一个File Footer，这里面存的是每个Stripe的行数，每个Column的数据类型信息等；每个文件的尾部是一个PostScript，这里面记录了整个文件的压缩类型以及FileFooter的长度信息等。在读取文件时，会seek到文件尾部读PostScript，从里面解析到File Footer长度，再读FileFooter，从里面解析到各个Stripe信息，再读各个Stripe，即从后往前读。

4.4 Parquet格式

Parquet文件是以二进制方式存储的，所以是不可以直接读取的，文件中包括该文件的数据和元数据，因此Parquet格式文件是自解析的。

1) 行组(Row Group)：每一个行组包含一定的行数，在一个HDFS文件中至少存储一个行组，类似于orc的stripe的概念。

2) 列块(Column Chunk)：在一个行组中每一列保存在一个列块中，行组中的所有列连续的存储在这个行组文件中。一个列块中的值都是相同类型的，不同的列块可能使用不同的算法进行压缩。

3) 页(Page)：每一个列块划分为多个页，一个页是最小的编码的单位，在同一个列块的不同页可能使用不同的编码方式。

通常情况下，在存储Parquet数据的时候会按照Block大小设置行组的大小，由于一般情况下每一个Mapper任务处理数据的最小单位是一个Block，这样可以把每一个行组由一个Mapper任务处理，增大任务执行并行度。