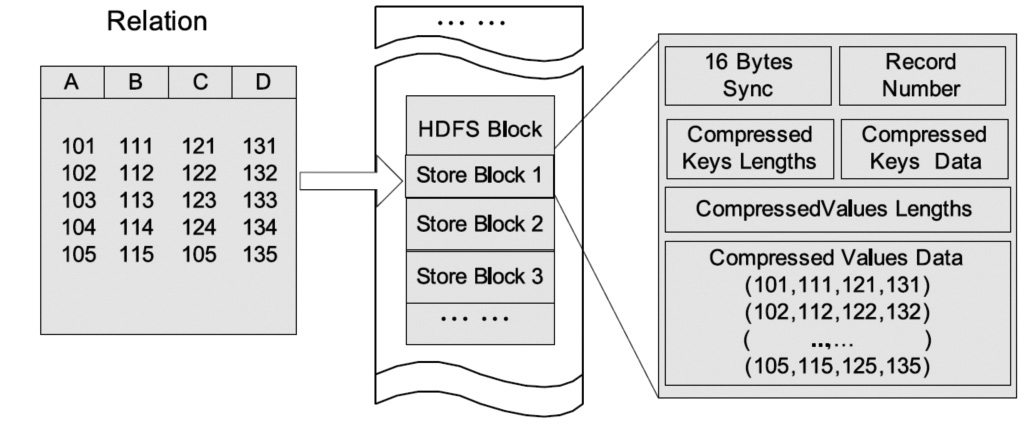
Hadoop常用文件格式

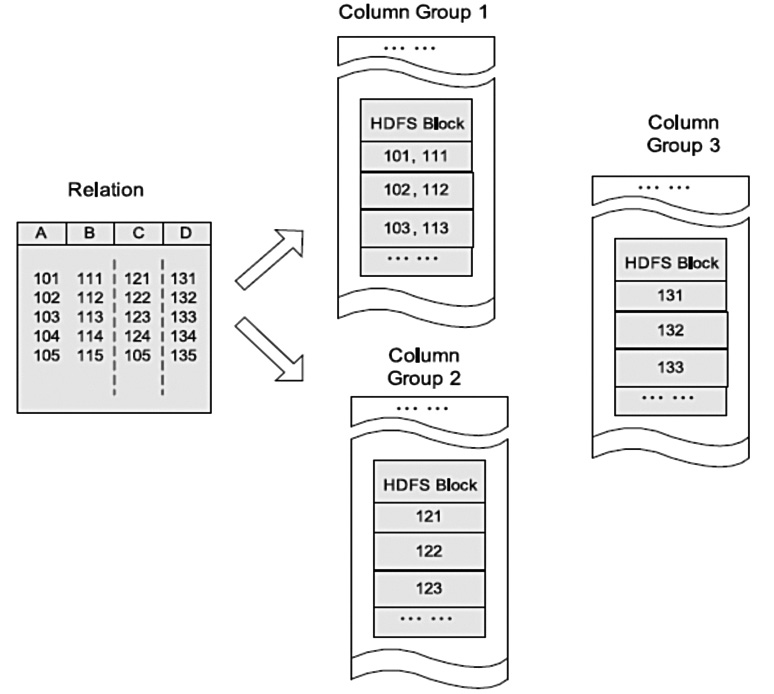
目前在Hadoop中，主要将数据的存储分成行存储和列存储两种文件格式

1. 行存储，以数据行为单位将数据聚簇在一起，其保存相同记录的所有域都在同一个集群节点，如下图：



但是查询少数列式，不能直接定位到列而跳过不需要的列。常见的面向行存储为SequenceFile、MapFile、Avro DataFile，采用这种方式，访问行的小部分数据，需要将整行读入内存，适用于整行数据需要同时处理的情况。

1. 列存储，以列数据集合为单位，一行记录会被拆分成多列，写入时追加到对应的列的末尾处，如下图：



在面向列的文件存储结构中，列A和列B存储在同一个列簇，而列C和列D分别存储在单独的列簇，这种结构使得在查询时能够直接读取所需的列而避免不必要列的读取，对相似的数据可以有一个更好的压缩比。可以充分利用列存储的特点支持向量计算，获取更好的扫描性能。常用的面向列存储的文件格式：Parquent、RCFile及ORCFile等。

Hadoop实际上支持任意文件格式，只要能够实现RecordWriter和RecordReader即可，其中数据库格式也会经常存在Hadoop中，比如HBase数据文件使用的数据格式为HFile。

# SequenceFile

SequenceFile是Hadoop的一个重要数据文件类型，将数据以<key,value>的形式序列化到文件中。这种二进制文件内部使用Hadoop标准化的Writable接口实现序列化和反序列化。在SequenceFile中有三种压缩状态，对应三种Writer

* Uncompressed，未进行压缩的状态，对应SequenceFile#Writer
* Record Compressed，对每条记录的Value值进行压缩，文件头中包含了使用哪种压缩算法，对应RecordCompressWriter
* Block Compressed，当数据量达到一定大小后，将停止写入进行整体压缩，其方式是把所有keyLength,key,vLength,value分别合在一起进行整体压缩。BlockCompressWriter

SequenceFile的格式由一个Header和跟随一个或多个Record组成，在SequenceFile中提供两种文件位置定位的方法：

* seek(long poisition)，posisition必须是记录的边界，否则调用next方法时会报错
* Sync(long poisition)，position可以不是记录的边界，如果不是边界会定位到下一个同步点，没有同步点，则跳转到文件的结尾位置。

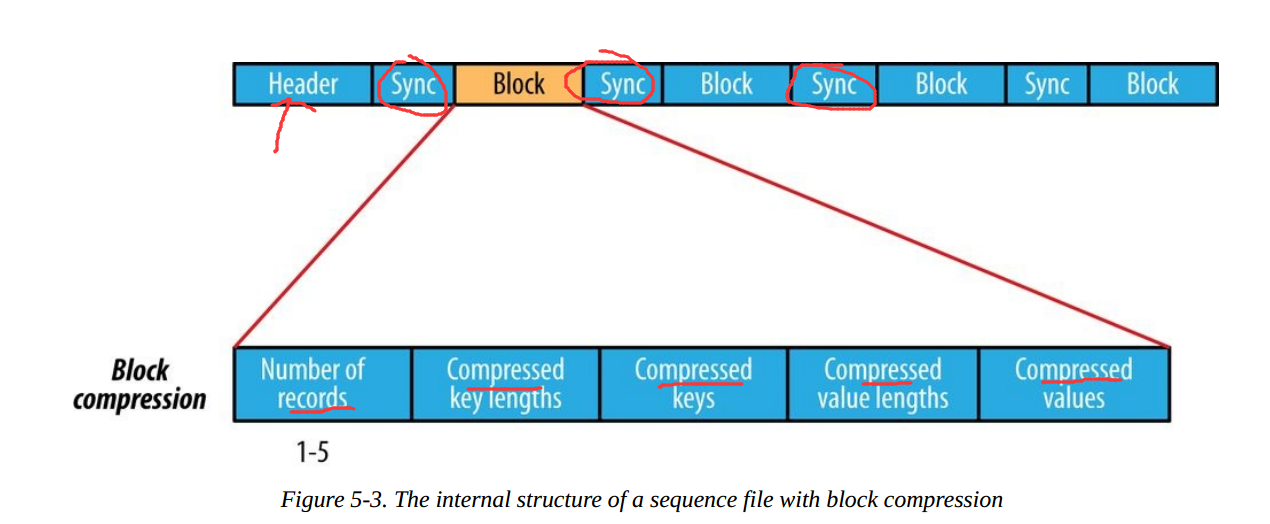
1. **Header**，每个SequenceFile都包含一个Header，包括以下几个部分：

* SEQ三个字母的byte数组
* Version Number的byte，目前为数字3的byte
* Key和Value的类名
* 压缩相关的信息
* 其他用户定义的元数据
* 同步标记，sync marker，16个字节

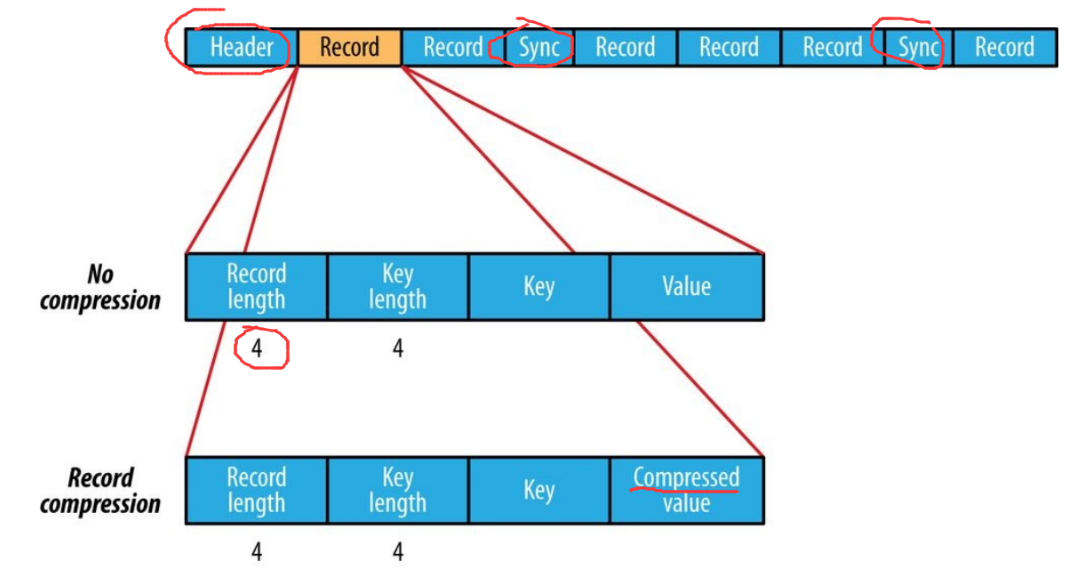
|  |  |
| --- | --- |
| IMG_256 | Metadata |

1. **Records，**在SequenceFile中每一个<key,value>被看做是一条Record，根据Record压缩策略，Record的写入格式也不同

* Block-Compressed格式，当数据量达到阈值后写入，所有的key压缩到一起为Key列表



* Record-Compressed格式，对Value进行压缩

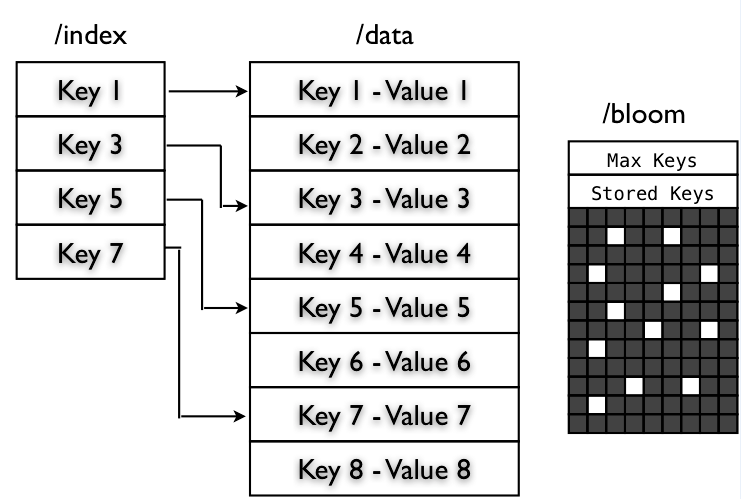


字节描述如下所示：

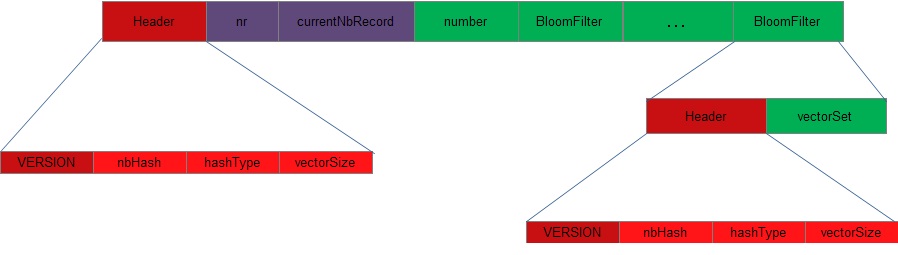
|  |  |
| --- | --- |
| IMG_256 | IMG_256 |

SequenceFile是Hadoop的一个基础数据文件，MapFile、SetFile、ArrayFile和BloomMapFile都是基于SequenceFile来实现的：

* MapFile，Key-Value对应的查找数据结构，由数据文件data和索引文件组成：数据文件包含所有需要存储的key-value对，按key顺序排序；索引文件包含一部分key值，用以指向数据文件的关键位置。在MapFile被访问的时候，索引文件会被加载到内存中，通过索引映射关系用以迅速定位到Record所在的文件位置，相对SequenceFile，检索效率更好。文件结构如下图：



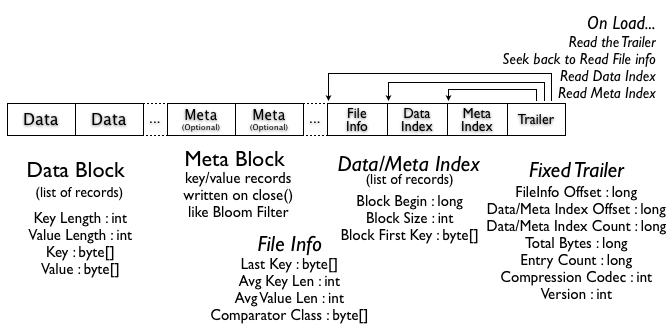
* SetFile，基于MapFile实现，其Value为NullWritable ，append操作仅存储Key
* ArrayFile，基于MapFile实现，其append操作仅写入value，Key为递增+1
* BloomMapFile，针对MapFile的get方法，使用动态Bloom过滤器进行优化，过滤器保存在内存中

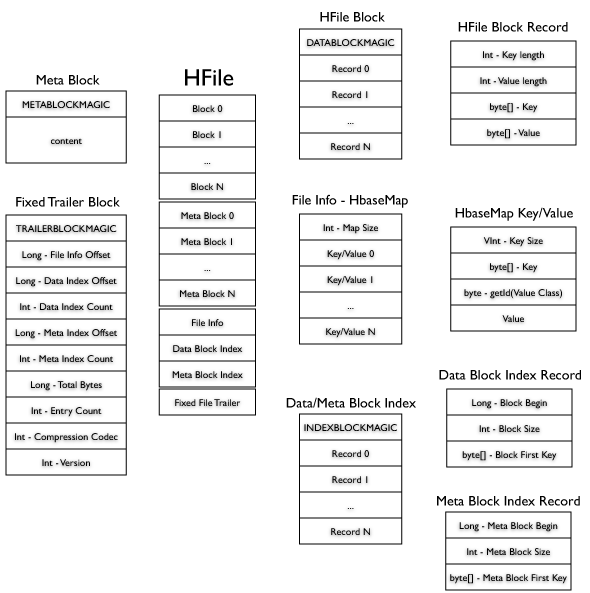


以上介绍的SequenceFile,MapFile,SetFile,ArrayFile及BloomMapFile是面向行的文件格式，在Hadoop中运行的MapReduce作业处理的大部分数据默认使用SequenceFile格式文件。

# HBase HFile&TFile

Apache HBase最初直接使用Hadoop的MapFile实现有序的Key存储，将数据和索引分别存储Hadoop SequencFile。但是由于每个文件都会分成数据和索引文件，进行读数据时可能需要扫描多个文件，为了减少需要读取的文件数，HBase将index和数据文件合并成一个文件，只对每个块的首个key做索引，同时增加metadata特性，文件结构如下所示：



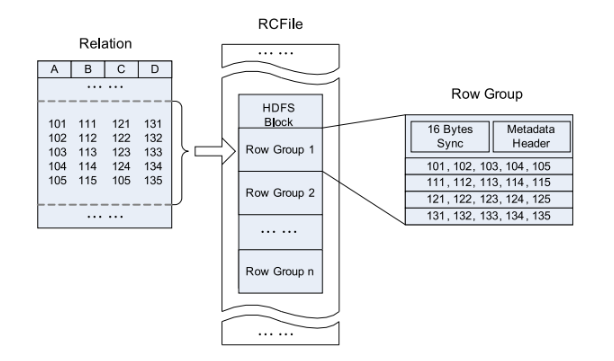


在HFile中存储一个列族的数据。

http://blog.cloudera.com/blog/2012/06/hbase-io-hfile-input-output/

# Hive RCFile&ORCFile

RCFile文件格式以Facebook开源的一种Hive文件存储格式，首先将表分为几个行组，对每个行组内的数据进行按类存储，每一列的数据都是分开存储，先水平划分，再垂直划分的理念，Hive使用RCFile进行数据存储，其在读取数据是对数据进行解压，而且针对特定的查询跳过不需要的列。文件结构如下图所示：

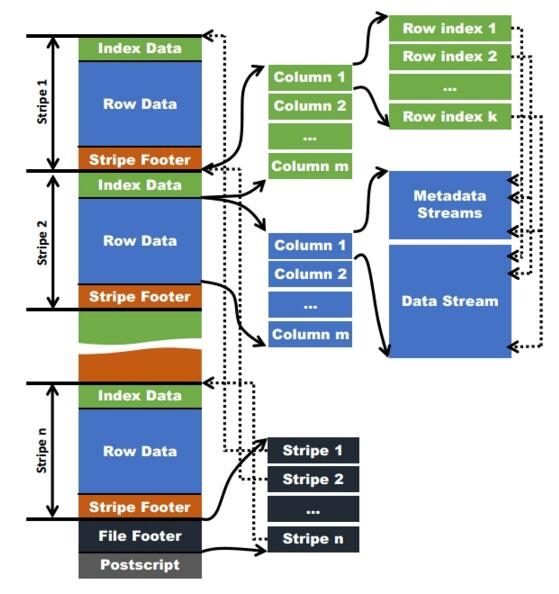


在行组中主要包括：

* 16字节的同步块信息，区分相邻行组
* 元数据的头部信息，包括该行组内的存储的行数、列的字段信息等

在RCFile中，将每一行存储为一列，一列存储为一行，当表很大，字段很多时只需要取出固定的一列即可。

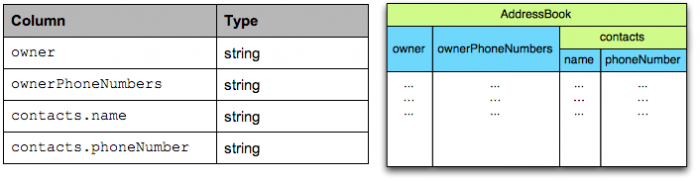
ORC是在一定程度上扩展了RCFile，在基础上引申出来的Stripe和Footer等，每个ORC文件首先会被横向切成多个Stripe，而每个Stripe内部已列存储，所有的列存储在一个文件中，文件结构如下图：



Stripe包含index、row和stripe footer，在Strip footer包含流位置的目录。在Index中包含每列的最大和最小值以及每列所在的行。行索引里面提供了偏移量，可以跳到正确的压缩块位置。

# Parquet

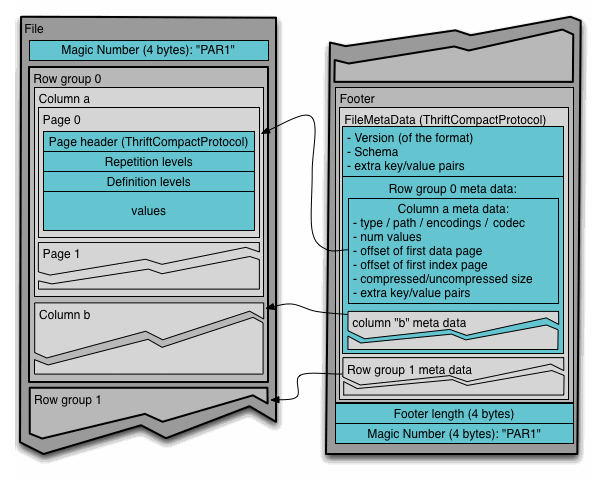
Parquet文件是以二进制的方式存储的，文件中包含该文件的数据和元数据，因此Parquet文件是自解析的。在Parquet格式存储中，Schema的叶子节点实际对应多少个Column，上图中实际上是4个Column，如下图所示：



前面已经介绍了Parquet文件的存储概念，下面是切分的模式

* 所有的数据被水平切分成Row Group，在Row Group中包含对应区间内所有列的column chunk，其是数据读写的缓存单元，设置较大的Group Goup可以增加并行度。
* Column Chunk负责存储某列的数据，这些数据包括：Repetition Levels，Definition Level和values
* Column Chunk由Page组成，Page是压缩和编码的单元，对数据模型来说是透明的。
* 一个Parquet文件的最后是Footer，存储文件的元数据信息和统计信息

对于AddressBook Schema为例，其形成的文件格式如下图所示：

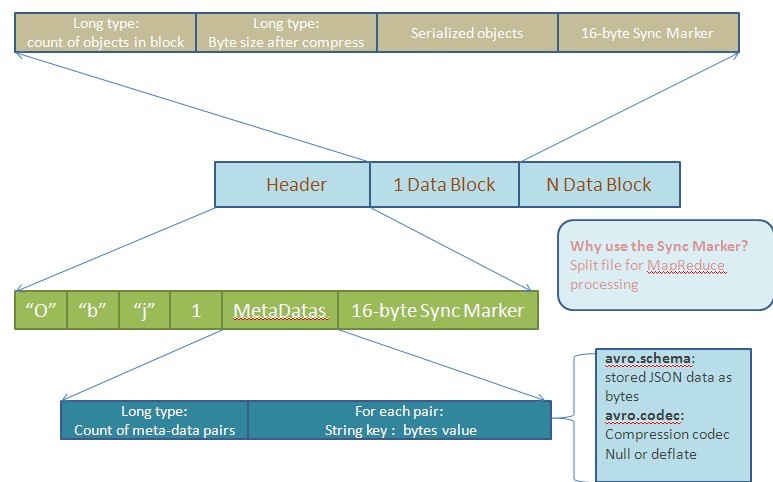


上图展示了一个Parquet文件的内容

* 一个文件可以存储多个行组
* 文件的首位是该文件的Magic Code，用于校验是否是一个Parquet文件
* Footer Length，文件元数据的大小，通过该值和文件长度可以计算元数据的便宜量
* FileMetadata，文件元数据包括一个行组的元数据信息和该文件存储数据的Schema信息
* 页元数据，除了文件中每一个行组的元数据，每一页的开始都会存储该页的元数据，在Parquet中，有三种类型的页:数据页、字典页和索引页。数据页用于存储当前行组中该列的值，字典页存储该列值的编码字段，每个列块中最多包含一个字典页，索引页用来存储当前行组下该列的索引。目前Parquet还不支持索引页。

# Apache Avro

Apache Avro是一种用于支持数据密集型的二进制文件格式，所有需要存储这个文件的对象都以二进制编码的形式写入，对象在文件中以块Block俩组织，并且这些对象都是可以被压缩的。块与块之间会存在同步标识符（Synchronization Marker），以便切割文件用于处理，文件结构图如下所示：



一个存储文件由两部分组成：头信息（Header）和数据块（Data Block）

1. 头信息

* 四个字节的前缀（Maic Numer）
* 文件的Meta-data，目前仅支持Schema和codec，codec表示对后面的文件数据块采用何种压缩方式（无压缩或者Deflate）。Schema是对象的定义字符串
* 同步标识块，使用该标识将大块数据分割成小块，连续存储在同一个文件中，便于并发处理

2）Data Block，Avro序列化对象

https://blog.csdn.net/yongjian\_luo/article/details/16809233

https://blog.csdn.net/garfielder007/article/details/50991729

https://blog.csdn.net/javastart/article/details/52739570

https://blog.csdn.net/u011812294/article/details/63262624

http://blog.cloudera.com/blog/2011/01/hadoop-io-sequence-map-set-array-bloommap-files/