| 序号 | 问题 | 答案 |
| --- | --- | --- |
| 1 | delete是如何删除数据的？ | 在执行delete操作的时候，不会立即删除，先打标记，直到当Store File合并时，再清除这些标记的文件。 |
| 2 | 列举flush的触发条件 | a. regionServer的全局memstore的大小超过阀值  b. 单个region里memstore的大小超过阀值  c. 单个memstore的大小超过阀值  d. WAL文件的数量超过阀值 |
| 3 | 比对minor compaction 和 major compaction的区别 | minor compaction操作只用来做部分文件的合并操作以及包括minVersion=0并且设置ttl的过期版本清理，不做任何删除数据和多版本数据的清理工作。  major compaction操作是对region下的store下的所有storeFile执行合并操作，最终的结果是整理合并出一个文件。 |
| 4 | WAL在HBase中的作用 | WAL提供了一种高并发、持久化的日志保存与回放机制。每一个业务数据的写入操作（Put / Delete）执行前，都会记录在WAL中。如果出现HBase服务器宕机，则可以从WAL中回放执行之前没有完成的操作 |
| 5 | 过多的列族有什么危害？ | 多个CF代表有多个Store，也就是说有多个MemStore，会导致内存的消耗量增大，flush的压力变大 |
| 6 | 描述读写前的寻址过程 | 1、访问ZK，确定hbase:meta表所在的root regionserver 2、访问root regionserver，通过hbase:meta来确定目标regionserver 3、直接访问目标regionserver进行实际的读写操作 |
| 7 | 简述写数据的流程 | 写入顺序：WAL -> memstore -> HFile |
| 8 | 简述读数据的流程 | 读取顺序：WAL -> blockcache -> HFile |
| 9 | rowkey的设计原则 | a. 满足业务的读取需要，复合rowkey中要把查询列前置 b. 避免写入热点，可以通过hash/反转等方式来实现 c. 尽量简短，因为rowkey和数据一样占据空间 |
| 10 | HMaster的作用 | a. 管理regionserver，对region进行分配 b. 负责DDL类操作 |
| 11 | HRegionServer的作用 | 负责具体的数据读写，regionserver本身不存放数据，数据落在HDFS中，通过HDFS的副本实现容错 |
| 12 | HBase数据分区的方式 | HBase中的数据是按rowkey顺序存放的，通过范围分区的方式把数据水平切分成不同的region |
| 13 | 列族和列的区别 | 列族是对列的分组，在建表时必须指定，而列是在写入数据时动态添加的，每个列必须属于某一个列族；HBase中很多可配置的参数都是作用在列族上的 |
| 14 | 如何理解HBase中的KeyValue对？ | Key = rowkey + cf:column + version，是个三维结构；Value是存放的具体数据，没有类型区别，都是字节数组；HBase中Table更像是SortedHashMap，而不像RDBMS中二维表格 |
| 15 | 列举下列族中的配置项 | 是否启用压缩，是否启用缓存，block的大小，TTL生命周期，最大版本数等 |
| 16 | 简述HBase在HDFS上的目录结构 | /hbase/<namespace>/<table>/<region\_id>/<cf>/<hfile文件> |
| 17 | 如何进行预分区？ | 1. 手动指定切分点 2. 根据目标分区数，通过算法自动决定切分点，如HexStringSplit |
| 18 | 过滤器的原理及作用 | a. 原理：谓词下推，服务器端过滤 b. 作用：辅助get/scan操作，减少传递到客户端的KV对 |
| 19 | 数据查询的方式有几种？ | a. Get : 单条记录查询 b. Scan : 指定范围查询 c. 全表扫描 |
| 20 | 描述HFile的结构 | HFile有不同的block构成，不同的block结构相同，内容有异，大致分为数据块、索引块、元数据块和bloomfilter块。查询时，先通过bloomfilter块来快速判断rowkey是否存在，进而通过索引块来定位记录 |