1. Hive的metastore的三种模式

- **内嵌Derby方式**

  这个是Hive默认的启动模式，一般用于单元测试，这种存储方式有一个缺点：在同一时间只能有一个进程连接使用数据库。

- **Local方式**

  本地MySQL

- **Remote方式**

  远程MySQL,一般常用此种方式

2. Hive 内部表和外部表的区别

- 建表时带有external关键字为外部表，否则为内部表
- 内部表和外部表建表时都可以自己指定location
- 删除表时，外部表不会删除对应的数据，只会删除元数据信息，内部表则会删除
- 其他用法是一样的

3. Hive 四种排序方式的区别

- **order by**
  order by 是要对输出的结果进行全局排序，这就意味着**只有一个reducer**才能实现（多个reducer无法保证全局有序）但是当数据量过大的时候，效率就很低。如果在严格模式下（hive.mapred.mode=strict）,则必须配合limit使用
- **sort by**
  sort by 不是全局排序，只是在进入到reducer之前完成排序，只保证了每个reducer中数据按照指定字段的有序性，是局部排序。配置mapred.reduce.tasks=[nums]可以对输出的数据执行归并排序。可以配合limit使用，提高性能
- **distribute by**
  distribute by 指的是按照指定的字段划分到不同的输出reduce文件中，和sort by一起使用时需要注意，distribute by必须放在前面
- **cluster by**

  cluster by 可以看做是一个特殊的distribute by+sort by，它具备二者的功能，但是只能实现倒序排序的方式,不能指定排序规则为asc 或者desc

4. Hive中大表join小表的优化方法

在小表和大表进行join时，将**小表放在前边**，效率会高，hive会将小表进行缓存

5. Hive中join都有哪些

Hive中除了支持和传统数据库中一样的内关联（JOIN）、左关联（LEFT JOIN）、右关联（RIGHT JOIN）、全关联（FULL JOIN），还支持左半关联（LEFT SEMI JOIN）

- **内关联（JOIN）**

  只返回能关联上的结果。

- **左外关联（LEFT [OUTER] JOIN）**

  以LEFT [OUTER] JOIN关键字前面的表作为主表，和其他表进行关联，返回记录和主表的记录数一致，关联不上的字段置为NULL。

- **右外关联（RIGHT [OUTER] JOIN）**

  和左外关联相反，以RIGTH [OUTER] JOIN关键词后面的表作为主表，和前面的表做关联，返回记录数和主表一致，关联不上的字段为NULL。

- **全外关联（FULL [OUTER] JOIN）**

  以两个表的记录为基准，返回两个表的记录去重之和，关联不上的字段为NULL。

- **LEFT SEMI JOIN**

  以LEFT SEMI JOIN关键字前面的表为主表，返回主表的KEY也在副表中的记录

- **笛卡尔积关联（CROSS JOIN）**

  返回两个表的笛卡尔积结果，不需要指定关联键。

6. Hive SQL 是怎样解析成 MR job 的?

主要分为6个阶段:

1. **Hive使用Antlr实现语法解析**.根据Antlr制定的SQL语法解析规则,完成SQL语句的词法/语法解析,将SQL转为抽象语法树AST.
2. **遍历AST,生成基本查询单元QueryBlock**.QueryBlock是一条SQL最基本的组成单元，包括三个部分：输入源，计算过程，输出.
3. **遍历QueryBlock,生成OperatorTree**.Hive最终生成的MapReduce任务，Map阶段和Reduce阶段均由OperatorTree组成。Operator就是在Map阶段或者Reduce阶段完成单一特定的操作。QueryBlock生成Operator Tree就是遍历上一个过程中生成的QB和QBParseInfo对象的保存语法的属性.
4. **优化OperatorTree.**大部分逻辑层优化器通过变换OperatorTree，合并操作符，达到减少MapReduce Job，减少shuffle数据量的目的
5. **OperatorTree生成MapReduce Job**.遍历OperatorTree,翻译成MR任务.
   - 对输出表生成MoveTask
   - 从OperatorTree的其中一个根节点向下深度优先遍历
   - ReduceSinkOperator标示Map/Reduce的界限，多个Job间的界限
   - 遍历其他根节点，遇过碰到JoinOperator合并MapReduceTask
   - 生成StatTask更新元数据
   - 剪断Map与Reduce间的Operator的关系
6. **优化任务.** 使用物理优化器对MR任务进行优化,生成最终执行任务

7. Hive UDF 简单介绍

在Hive中，用户可以自定义一些函数，用于扩展HiveQL的功能，而这类函数叫做UDF（用户自定义函数）。UDF分为两大类：UDAF（用户自定义聚合函数）和UDTF（用户自定义表生成函数）。

Hive有两个不同的接口编写UDF程序。一个是基础的UDF接口，一个是复杂的GenericUDF接口。

1. org.apache.hadoop.hive.ql. exec.UDF 基础UDF的函数读取和返回基本类型，即Hadoop和Hive的基本类型。如，Text、IntWritable、LongWritable、DoubleWritable等。
2. org.apache.hadoop.hive.ql.udf.generic.GenericUDF 复杂的GenericUDF可以处理Map、List、Set类型。

8. HMaster宕机的时候,哪些操作还能正常工作

对表内数据的增删查改是可以正常进行的,因为hbase client 访问数据只需要通过 zookeeper 来找到 rowkey 的具体 region 位置即可。

但是对于创建表/删除表等的操作就无法进行了，因为这时候是需要HMaster介入，并且region的拆分、合并、迁移等操作也都无法进行了。

9. Impala 和 hive 的查询有哪些区别

Impala是基于Hive的大数据实时分析查询引擎，直接使用Hive的元数据库Metadata,意味着impala元数据都存储在Hive的metastore中。并且impala兼容Hive的sql解析，实现了Hive的SQL语义的子集，功能还在不断的完善中。

Impala相对于Hive所使用的优化技术

- 1、没有使用 MapReduce进行并行计算，虽然MapReduce是非常好的并行计算框架，但它更多的面向批处理模式，而不是面向交互式的SQL执行。与 MapReduce相比：Impala把整个查询分成一执行计划树，而不是一连串的MapReduce任务，在分发执行计划后，Impala使用拉式获取 数据的方式获取结果，把结果数据组成按执行树流式传递汇集，减少的了把中间结果写入磁盘的步骤，再从磁盘读取数据的开销。Impala使用服务的方式避免 每次执行查询都需要启动的开销，即相比Hive没了MapReduce启动时间。
- 2、使用LLVM产生运行代码，针对特定查询生成特定代码，同时使用Inline的方式减少函数调用的开销，加快执行效率。
- 3、充分利用可用的硬件指令（SSE4.2）。
- 4、更好的IO调度，Impala知道数据块所在的磁盘位置能够更好的利用多磁盘的优势，同时Impala支持直接数据块读取和本地代码计算checksum。
- 5、通过选择合适的数据存储格式可以得到最好的性能（Impala支持多种存储格式）。
- 6、最大使用内存，中间结果不写磁盘，及时通过网络以stream的方式传递。

注：资料来源于网络。