4、Spark SQL优化策略

查询优化是传统数据库中最为重要的一环，这项技术在传统数据库中已经很成熟。除了查询优化，Spark SQL在存储上也进行了优化下面介绍Spark SQL的一些优化策略。

（1）内存列式存储与内存缓存表

Spark SQL可以通过cacheTable将数据存储转换为列式存储，同时将数据加载到内存缓存。cacheTable相当于在分布式集群的内存物化视图，将数据缓存，这样迭代的或者交互式的查询不用再从HDFS读数据，直接从内存读取数据大大减少了I/O开销。列式存储的优势在于Spark SQL只需要读出用户需要的列，而不需要像行存储那样每次都将所有列读出，从而大大减少内存缓存数据量，更高效地利用内存数据缓存，同时减少网络传输和I/O开销。数据按照列式存储，由于是数据类型相同的数据连续存储，所以能够利用序列化和压缩减少内存空间的占用。

（2）列存储压缩

为了减少内存和硬盘空间占用，Spark SQL采用了一些压缩策略对内存列存储数据进行压缩。Spark SQL的压缩方式要比Shark丰富很多，如它支持PassThrough、RunLengthEncoding、DictionaryEncoding、BooleanBitSet、IntDelta、LongDelta等多种压缩方式，这样能够大幅度减少内存空间占用、网络传输和I/O开销。

（3）逻辑查询优化

SparkSQL在逻辑查询优化（见图8-4）上支持列剪枝、谓词下压、属性合并等逻辑查询优化方法。列剪枝为了减少读取不必要的属性列、减少数据传输和计算开销，在查询优化器进行转换的过程中会优化列剪枝。

下面介绍一个逻辑优化的例子。

SELECT Class FROM （SELECT ID，Name，Class FROM STUDENT ） S WHERE S.ID=1

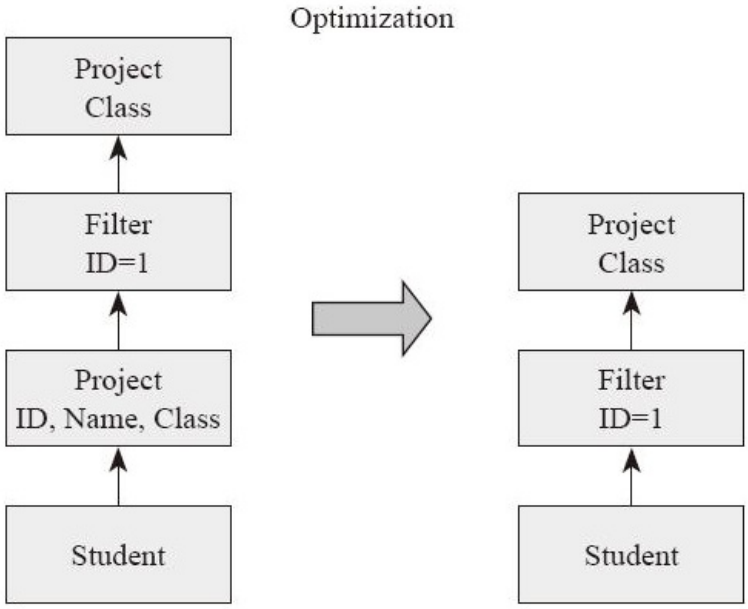

图8-4 逻辑查询优化

Catalyst将原有查询通过谓词下压，将选择操作ID=1优先执行，这样过滤大部分数据，通过属性合并将最后的投影只做一次，最终保留Class属性列。

（4）Join优化

Spark SQL深度借鉴传统数据库的查询优化技术的精髓，同时在分布式环境下调整和创新特定的优化策略。现在Spark SQL对Join进行了优化，支持多种连接算法，现在的连接算法已经比Shark丰富，而且很多原来Shark的元素也逐步迁移过来，如BroadcastHashJoin、BroadcastNestedLoopJoin、HashJoin、LeftSemiJoin，等等。

下面介绍其中的一个Join算法。

BroadcastHashJoin将小表转化为广播变量进行广播，这样避免Shuffle开销，最后在分区内做Hash连接。这里使用的就是Hive中Map Side Join的思想，同时使用DBMS中的Hash连接算法做连接。 随着Spark SQL的发展，未来会有更多的查询优化策略加入进来，同时后续Spark SQL会支持像Shark Server一样的服务端和JDBC接口，兼容更多的持久化层，如NoSQL、传统的DBMS等。一个强有力的结构化大数据查询引擎正在崛起。