Spark读取数据库(Mysql)的四种方式讲解

目前Spark支持四种方式从数据库中读取数据，这里以Mysql为例进行介绍。

一、不指定查询条件

这个方式链接MySql的函数原型是：

```
def jdbc(url: String, table: String, properties: Properties): DataFrame
```

我们只需要提供Driver的url，需要查询的表名，以及连接表相关属性properties。下面是具体例子：

```
val url = "jdbc:mysql://www.iteblog.com:3306/iteblog?user=iteblog&password=iteblog"

val prop = new Properties()
val df = sqlContext.read.jdbc(url, "iteblog", prop )

println(df.count())
println(df.rdd.partitions.size)
```

我们运行上面的程序，可以看到df.rdd.partitions.size输出结果是1，这个结果的含义是iteblog表的所有数据都是由RDD的一个分区处理的，所以说，如果你这个表很大，很可能会出现OOM

```
WARN TaskSetManager: Lost task 0.0 in stage 1.0 (TID 14, spark047219):
 java.lang.OutOfMemoryError: GC overhead limit exceeded at com.mysql.jdbc.MysqlIO.reuseA
ndReadPacket(MysqlIO.java:3380)
```

这种方式在数据量大的时候不建议使用。

过往记忆博客：http://www.iteblog.com

如果想及时了解Spark、Hadoop或者Hbase相关的文章，欢迎关注微信公共帐号：iteblog_hadoop

二、指定数据库字段的范围

这种方式就是通过指定数据库中某个字段的范围，但是遗憾的是，这个字段必须是数字，来看看这个函数的函数原型：

```
def jdbc(
    url: String,
    table: String,
    columnName: String,
    lowerBound: Long,
    upperBound: Long,
    numPartitions: Int,
    connectionProperties: Properties): DataFrame
```

前两个字段的含义和方法一类似。columnName就是需要分区的字段，这个字段在数据库中的类型必须是数字；lowerBound就是分区的下界；upperBound就是分区的上界；numPartitions是分区的个数。同样，我们也来看看如何使用：

```
val lowerBound = 1
val upperBound = 100000
val numPartitions = 5
val url = "jdbc:mysql://www.iteblog.com:3306/iteblog?user=iteblog&password=iteblog"

val prop = new Properties()
val df = sqlContext.read.jdbc(url, "iteblog", "id", lowerBound, upperBound, numPartitions, pro
```

p)

这个方法可以将iteblog表的数据分布到RDD的几个分区中，分区的数量由numPartitions参数决定，在理想情况下，每个分区处理相同数量的数据，我们在使用的时候不建议将这个值设置的比较大，因为这可能导致数据库挂掉！但是根据前面介绍，这个函数的缺点就是只能使用整形数据字段作为分区关键字。

这个函数在极端情况下，也就是设置将numPartitions设置为1，其含义和第一种方式一致。

三、根据任意字段进行分区

基于前面两种方法的限制，Spark还提供了根据任意字段进行分区的方法，函数原型如下：

```
def jdbc(
    url: String,
    table: String,
    predicates: Array[String],
    connectionProperties: Properties): DataFrame
```

这个函数相比第一种方式多了predicates参数，我们可以通过这个参数设置分区的依据，来看看例子：

```
val predicates = Array[String]("reportDate <= '2014-12-31'",
 "reportDate > '2014-12-31' and reportDate <= '2015-12-31'")
val url = "jdbc:mysql://www.iteblog.com:3306/iteblog?user=iteblog&password=iteblog"

val prop = new Properties()
val df = sqlContext.read.jdbc(url, "iteblog", predicates, prop)
```

最后rdd的分区数量就等于predicates.length。

四、通过load获取

Spark还提供通过load的方式来读取数据。

```
sqlContext.read.format("jdbc").options(
```

```
Map("url" -> "jdbc:mysql://www.iteblog.com:3306/iteblog?user=iteblog&password=iteblog",
  "dbtable" -> "iteblog")).load()
```

options函数支持url、driver、dbtable、partitionColumn、lowerBound、upperBound以及numPartitions选项，细心的同学肯定发现这个和方法二的参数一致。是的，其内部实现原理部分和方法二大体一致。同时load方法还支持json、orc等数据源的读取。