1、Overview

We propose a novel framework to accelerate complex SPARQL queries by summary-based vertical partitioning and statistics optimization over the Spark platform. We introduce vertical partitioning approach for RDF summary graph called SumVP, and complex SPARQL queries are made by translating them into Spark SQL queries based on binary statistics of tables (BT) and ternary tables (TT) to accelerate querying evaluation. We use Spark SQL API to execute query evaluation.

2、Requirements

Java 8+

Scala 2.11+

Apache Hadoop 2.6+

Apache Spark 2.4+

IntelliJ IDEA 2019+

Apache Maven 3.9+

3、How to use it?

In this package, we provide sources to process and query RDF datasets. For space reasons, we did not unload the datasets. You can get the corresponding RDF datasets from LUBM, WatDiv and other places. It is worth noting that all RDF datasets have to be in the N-Triples format. If your RDF data is in XML format, you can use (rdf2rdf-1.0.1-2.3.1.jar) to convert it to NT format. There are four modules in SumVP. In each module, we provide a running jar package. Of course, you can also run our code in your own way.

（1）graph summarization(GraphSum)

**DESCRIPTION：**

GraphSum is the module used to process RDF data into summary graph based on hashtable schema. You should have an RDF dataset in NT format as program input. The output of program is a summary graph hashtable. GraphSum is a maven project. You need to install maven to compile the project.

**INSTALL:**

cd GraphSum

mvn package

**EXECUTION:**

spark submit

（2）BT/TT create (DataCreator)

**DESCRIPTION：**

The function of this module is to convert the hash table of the summary graph generated in the graph summarization module into the binary table and ternary table. During the creation process DataCreator generates four statistic files (stat\_vp.txt, stat\_ss.txt, stat\_os.txt, stat\_so.txt), which are used for SPARQL query translation to SQL by QueryTranslator.

**EXECUTION:**

spark-submit --class dataCreator.runDriver --master <master-url> --deploy-mode <deploy-mode> --conf <key>=<value> dataCreator.jar <HDFS path> <RDF file>

（3）SPARQL query translation (SPARQLTranslator)

**DESCRIPTION：**

The purpose of SPARQLTranslator is to optimize SPARQL queries according to the statistical files generated in the DataCreator module and translate the optimized queries into spark SQL queries. SPARQLTranslator supports batch translation, which allows you to input a query path, so that SPARQLTranslator can translate all SPARQL queries under this path into corresponding spark SQL. SPARQLTranslator is an IntelliJ IDEA project. You need to install IntelliJ IDEA to compile the project.

**EXECUTION：**

spark-submit --class sparqlTranslator.runDriver --master <master-url> --deploy-mode <deploy-mode> --conf <key>=<value> SPARQLTranslator.jar <hashtable path> <SPARQL query path> <statistics path> <Spark SQL path>

（4）query execute (QueryExecutor)

**DESCRIPTION：**

QueryExecutor module submits the Spark SQL query generated by the SPARQLTranslator module to spark for execution. QueryExecutor allows you to submit spark SQL queries translated from multiple SPARQL queries at the same time, and no need to reload data. QueryExecutor is a maven project. You need to install Maven to compile this project. Of course, you can also compile the project in your own way.

**EXECUTION：**

spark-submit --class queryExecutor.runDriver --master spark://master:7077 --master <master-url> --deploy-mode <deploy-mode> --conf <key>=<value> QueryExecutor.jar <HDFS path> <hashtable path> <query file>