Master Thesis
On
Sparql On GraphFrames

Under the SuperVision
of:
Dr.Muhammad Abulaish

Submitted to:
Department of Computer Science
Faculty of Mathematics and Computer Science
South Asian University, New Delhi - 110021, India

Submitted by:
Ramazan Ali

Topic Name

Sparql On GraphFrames

Master Thesis
Computer Science
By:

Ramazan Ali

Under the Supervision Of
Dr.Muhammad Abulaish

Submitted in partial fulfillment of the requirements for the degree of
Master of Science,
Computer Science

Department of Computer Science
Faculty of Mathematics and Computer
Science
South Asian University, New Delhi, India

May 2017

Faculty of Mathematics and Computer Science
South Asian University

The master thesis entitled "Sparql On GraphFrames" being submitted in the partial fulfillment of the requirement for the award of the degree of M.Sc. (Computer Science), is a record of original and bona-de work carried out by the undersigned in Faculty of Mathematics and Computer Science, South Asian University, New Delhi, India. The work presented in this thesis has not been submitted to any other University or Institute for the award of any degree or diploma.

Supervisor
Depp of Computer science
Faculty of Mathematics and Computer Scienc
South Asian University
New Delhi, India.

Chair Person
Depp of Computer science
Faculty of Mathematics and Computer Scienc
South Asian University
New Delhi, India.

Date: May 2017

Abstract

RDF (resource description framework) is meta-data data model and a W3C standard. It facilitates a human-readable and a machine-processable linked data. Recently RDF has been used widely in many different domains and as a result large amount of RDF data is being generated and becoming available which is beyond the processing power of traditional standalone machines . Distributed cluster computing frameworks like spark and Hadoop being the current solution for scalable management of big data, is a good option for managing big RDF datasets. since RDF data is modeled as a directed, labeled graph; the graph processing capability of the above mentioned frameworks could be an efficient solution for processing large rdf data. in this document we will review the state of Art solutions for running Sparql queries on large datasets, and then we use motif finding in GraphFrames, one of the most recent system for graph processing based on spark, to run Sparql queries. We run 12 of the 14 standard LUBM bench mark queries and then compare our work with the previous work Spar(k)ql [8] which is on graphX, the other API of spark for graph processing. Our result shows that as the data size grows , the motif finding approach in graphFrames, which uses dataframes and joins, performs better in comparison to Spar(k)ql message passing method on graphX.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

1.1 Overview

The Idea of making statements about the resource has inspired the RDF data model. This model is in the form of triplets(subject-predicate-Object).Subject represents the resource, Object could be another resource or a property of the resource and predicate is the relationship between subject and object ,and it represents an aspect of the subject. RDF data has several serialization formats, among them the Ntriple being the human friendlly and easy to read by eye. The Ntriple format is a series of triples, each triple consisting of (subject-predicate-Object). Here is the Informal representation of triples and the coresponding visualized graph (see (Fig. 1.1)).

```
Aprofessor0-IsAdvisoreOf-GStudent0
AProfessor0-Teaches-GCourse0
GStudent0-TakesCourse-GCourse0
GStudent0-EmailAddress-''GraduateStudent0@department1.university0.edu''
```

GraphFrames, the API of spark for graph processing consists of two DataFrames, vertices and edges. Vertices could be any DataFrames consisting of number of columns but one being unique, to uniquely identify the node and Edges consists of three columns , the source, the destination and the label. Source and destination columns are the unique IDs in the source and destination nodes respectively. We create DataFrames for vertices and edges and then create the GraphFrames graph from the RDF data. Once the graph is created, we can use the full capability of spark and its APIs to process the graph. Specifically For evaluating the SPARQL queries ,we used motif finding, which refers to searching for structural patterns in the graph [25].

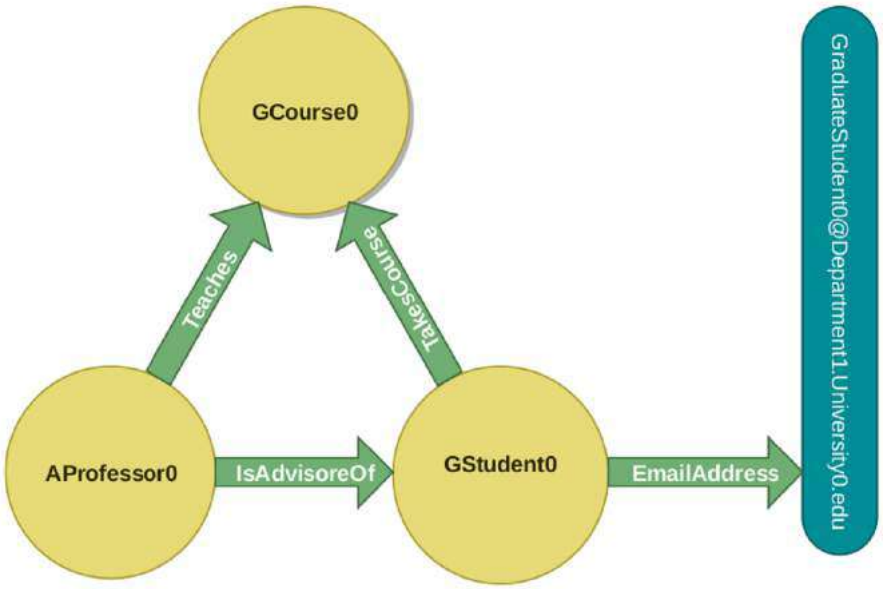


Fig. 1.1 Graph representatin

1.2 Motivation

Cluster computing frameworks like spark and Hadoop is a fine option for big data. Especially spark is well suited for iterative algorithms as it keep the data in memory and the swapping overhead of moving data in and out of memory is removed. In this project we have chosen spark as solution for big RDF graphs. Spark has two APIs for graph processing, GraphX and GraphFrames. GraphX have been previously used for SPARQL queries, but GraphFrames being new, has not been tested for SPARQL queries, and to the best of our knowledge we are the first to use it for SPARQL queries. The efficiency of DataFrames , on which GraphFrames is based and the optimized joins makes it good choice for processing big Graphs.

1.3 Problem Statement

With the widespread use of RDF in diverse application domains, a huge amount of RDF data is being proliferated and becoming available. As a result, an efficient and scalable management of RDF data is of increasing importance [4] . Cluster computing frameworks is the solution for ever increasing big data problem. Among these frameworks Apache spark is especially suitable for iterative algorithms. It has two APIs for graph processing, namely graphX and Graphframes. GraphFrames provide a concise, declarative API based on the “data frame” concept in R that can be used for both interactive queries and standalone programs. Under this API, GraphFrames use a graph-aware join optimization algorithm

across the whole computation that can select from the available views [3]. GraphFrames is implemented over Spark SQL, enabling parallel execution on Spark and integration with custom code [3]. In addition, GraphFrames's view abstraction makes it easy to further speed up interactive queries by registering the appropriate view, and that the combination of graph and relational data allows for other optimizations, such as attribute-aware partitioning.[2] since GraphX has already been tested for SPARQL queries, we felt the need to evaluate the much more optimized GraphFrames for running SPARQL queries.

1.4 Thesis Outline

The organizatin of remaining chapters of this thesis is as below:
Chapter2: In this chapter the topic of th my thesis, SPARQL on GraphFrames, is elaborated. Spacifically the cluster computing framework of GraphFrame is introduced. The Resource Description Framework is discussed and its query language , SPARQL, is talked about.
Chapter3: In this chapter the state of Art literature of cluster computing frameworks used for SPARQL is discussed. The current related literature is categorized in three main catergory namely 1) Hadoop based approach 2) Spark based and 3) vertix centric programming approach. Also in this chapter the literature on efficiency of DataFrames vs RDDS is reviewed.
Chapter4: Chapter 4 is about the methodology I have used to evaluate SPARQL. Mainly in this chapter I have described how the data is generated and converted to the desired format, how the GraphFame graph is created from the data and then the exact steps taken to answer the query is discussed.
Chapter5: In this chapter the performance of the system is compared with the current state of Art approaches, and it is shown how our approach save its integrity and reliability as the data size grow. Chapter 6: In this chapter I have concluded my thesis.

Chapter 2

SPARQL Queries over GraphFrames

2.1 GraphFrames

GraphFrames is one of the two graph processing API of Spark, the other being graphX.GraphFrames is implemented over Spark SQL, enabling parallel execution on Spark and easy integration with Spark's external data sources, built-in libraries, and custom ETL code [3] . GraphFrames makes it easy to write complete graph processing pipelines and enable optimizations across them that are not possible in current systems [3]. GraphFrames has kept the competencies present in GraphX , and by making efficient use of DataFrames, it has added even more functionalities for graph query, which is not present in graphX e.g Motif finding and DataFrame-based serialization.

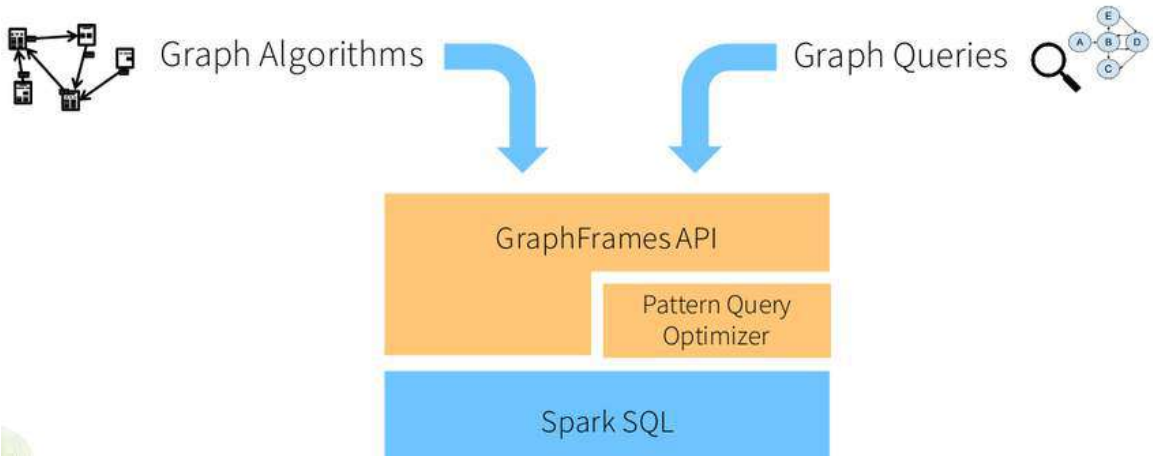


Fig. 2.1 (source:Spark-Summit 2016 , data science and engineering at scale.)

2.2 Resource Description Framework(RDF)

The Resource Description Framework (RDF) is a framework for representing information resources on the web, which is proposed by W3C(world Wide Web Consortium) as a recommendation(Manola and miller, 2004) [4].It has features that facilitate data merging even if the underlying schema differ, and it specifically supports the evolution of schema over time without requiring all the data consumers to be changed [19]. RDF can represent both structured and non-structured data, and more importantly metadata (the data that specify semantic information about the data) of resources on the web represented by RDF can be shared and exchanged among application programming without semantic missing [4]. This has made RDF widely used and popular and many organizations, companies and enterprises have started using rdf for representing and processing their data [4]. Among them are united states(data.gov), New York Times, BBC, and Best Buy ,RDF is finding increasing use in a wide range of web data management scenarios [4].

2.2.1 RDF Representation

RDF being the metadata about WWW resources, is built around resources with URI(Uniform Resource Identifier),used to uniquely identify resources. The resource description framework (RDF) is an abstract concept and can be modeled using different serializations methods, each of which has pros and cons.

- Turtle,allows a compact form [21].
- N-Triples is consisting of triples subject-predicate-object [20].
- N-Quads [18].
- JSON-LD Json based representation of RDF [JSON].
- Notation3/N3.
- RDF/XML , the XML based representation of RDF data [22].

Mostly RDF is expressed in XML/RDF format. Support for XML and that namespaces can be used instead of full URI, which itself reduces data size is the reason for popularity of this format . XML format is difficult to be read by eye.Turtle format however is the human friendly serialization method and can easily be read by human eye. The Ntriple format which is even simpler than Turtle is consisting of triples separated by dots. From the programming point of view, ntriple is easy to parse and is the format used for building DataFrames.
The following triples is illustrated as a graph.

(<http://spark.elte.hu#Aprofessor0>-<http://spark.elte.hu#IsAdvisoreOf>
<http://spark.elte.hu#GStudent0>)
(<http://spark.elte.hu#AProfessor0>-<http://spark.elte.hu#Teaches>-
<http://spark.elte.hu#GCourse0>)
(<http://spark.elte.hu#GStudent0>-<http://spark.elte.hu#TakesCourse>-
<http://spark.elte.hu#GCourse0>)
(<http://spark.elte.hu#GStudent0>-<http://spark.elte.hu#EmailAddress>-
''GraduateStudent0@department1.university0.edu'')

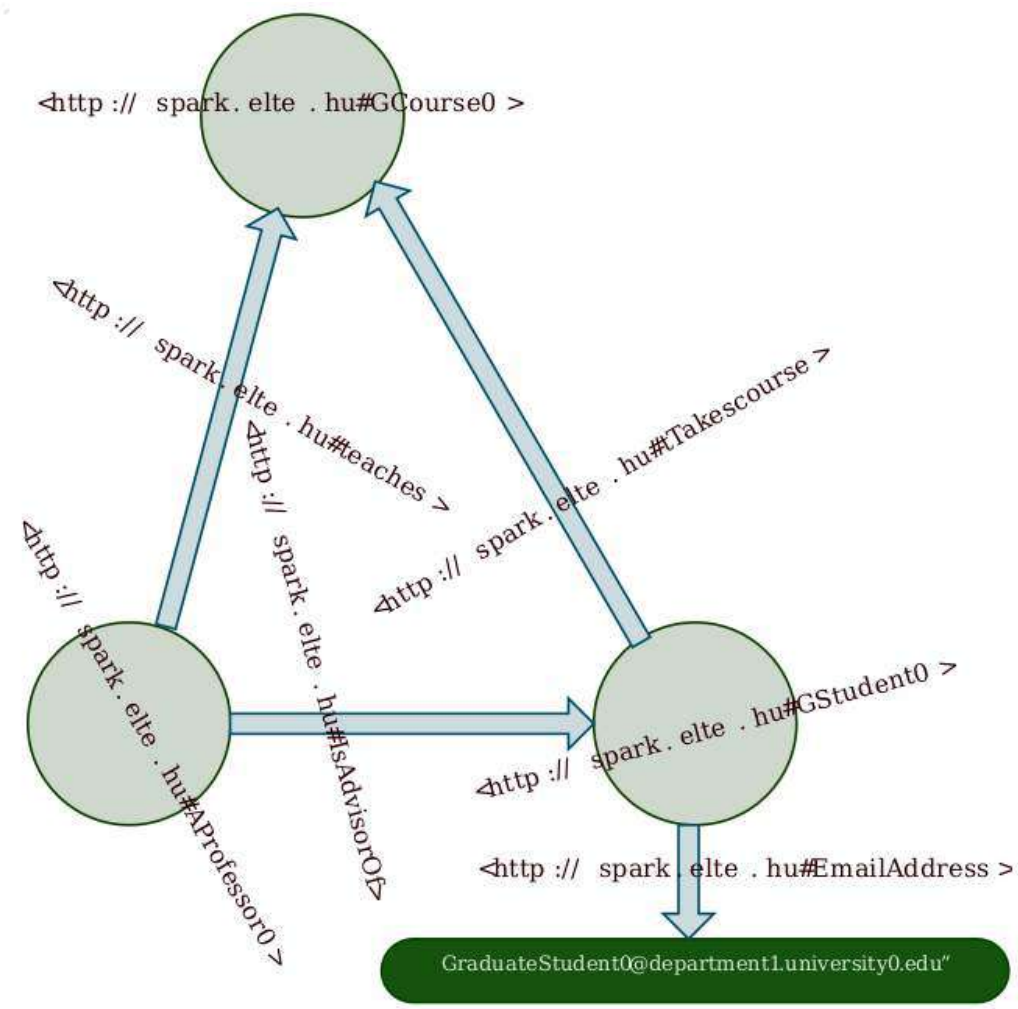
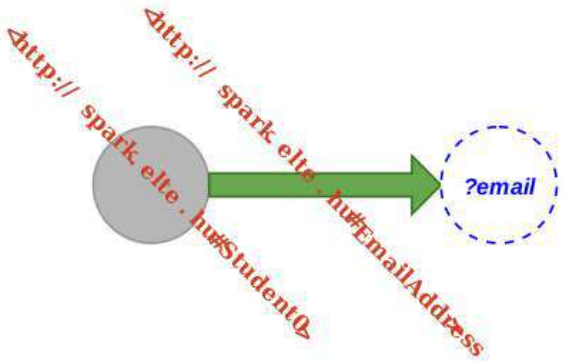


Fig. 2.2 Triples visualized in Graph

```
SELECT ?email
WHERE
{
<http://spark.elte.hu#Student0>
<http://spark.elte.hu#EmailAddress> ?email .
}
```

(a) An SPARQL query to select email of a student



(b) The query as a graph

email
GraduateStudent0@department1.university0.edu

(c) Query answer

Fig. 2.3 SPARQL query as graph and its answer

2.3 SPARQL

SPARQL (pronounced "sparkle", a recursive acronym [5] for SPARQL Protocol and RDF Query Language) is an RDF query language, that is, a semantic query language for databases, able to retrieve and manipulate data stored in Resource Description Framework (RDF) format [12, 24]. In particular it is a directed, labeled graph data format for representing information in the Web [6]. RDF is often used to represent, among other things, personal information, social networks, metadata about digital artifacts, as well as to provide a means of integration over disparate sources of information [26]. This specification defines the syntax and semantics of the SPARQL query language for RDF [26].

2.4 Datasets and DataFrame

Dataset is a new API in Spark. It is defined as collection of distributed data, that has both the beniefit of RDDs (strong typing, ability to use powerful lambda functions) and Spark SQL's optimized execution engine [25] . A DataFrame in spark is dataset arranged in named columns, conceptually equivalent of a table in relational database. It has been inspired by DataFrame that is in R and Python (panda).Unlike R and Python , DataFrame in spark is much more efficient as spark has automatic query optimizer for its DataFrame. Before any computation on a DataFrame starts, the Catalyst optimizer compiles the operations that were used to build the DataFrame into a physical plan for execution. Because the optimizer understands the semantics of operations and structure of the data, it can make intelligent decisions to speed up computation.[28]. Once built, DataFrames provide a domain-specific language for distributed data manipulation [23] .

Chapter 3

Literature review

There are RDBS system has been used for RDF data management. But the limitations with RDBMS is being on a single Machine. Single machines however lack the scalability and therefore can not meet the excessive growth of RDF data. As a result some researchers have used cluster computing frameworks for processing RDF data. In this chapter SPARQL query answering techniques based on cluster computing frameworks are categorized into three. Those based on hadoop mapReduce , those based on apache spark and the vertex centric programming technique. Though graphX is an API of spark, but we have put them under vertex centric based approaches due to its vertex centric characteristics.

3.1 SPARQL Query using Hadoop and MapReduce

The most prominent cluster computing framework, Hadoop and MapReduce has been explored to answer SPARQL queries . In this paper [16] they have worked both on storage and retrieval of RDF data using MapReduce. They store the RDF data in hadoop distributed file system(HDFS), and has developed an algorithm that by using mapreduce answers the SPARQL queries. Mapreduce tasks consist of series of map and reduce jobs in a way that Each mapreduce task loads the data into memory and then after processing it, stores it back to the disk. That is why the disk access time increases dramatically when it comes to iterative algorithms, and graph related algorithms are iterative in nature. To overcome this limitation of MapReduce, at one hand the other API of spark is being tested for graph processing, and in the other hand apache spark is specifically developed to overcome the disk access time. Spark keeps the data in the memory as long as it needs and once the processing is over, it stores the result back into the disk.

Hadoop MapReduce with NoSQL distributed data store [21]have used Join algorithms that execute joins according to query selectivity to reduce processing; and adaptive choice among

centralized and distributed (MapReduce-based) join execution for fast query responses [17]. There are also frameworks that work with hadoop , but use different query language than that of hadoop e.g pig latin and SQL. So some has used these frameworks too for running SPARQL queries. Sempala [16] has tried to overcome the limitation of data-intensive characteristics of hadoop based approaches, and they have used SPARQL-over-SQL-on-Hadoop to favor the selective nature of SPARQL queries. PigSPARQL [2] process SPARQL queries on Mapreduce cluster, but first queries are translated into pig Latin, the language developed by Yahoo for data analysis.

3.2 SPARQL using Spark

"SPARQL Query Processing with Apache Spark" [8] explores and compares five different query processing approaches, based on different join execution models and Spark components .Interesting to our work ,one of this approaches is based on DataFrame(DF) component of spark. They have shown that DF would be a good option when the size of RDDs is close to saturation in the main-memory of the cluster [8].

Method	Co-partitioning	Join algorithm	Merged access	Query optimization	Data compression
SPARQL RDD	✓	*Pjoin*	✗	✗	✗
SPARQL DF	✗(≤ v1.5)	*Pjoin, Brjoin*1	✗	poor	✓
SPARQL SQL	✗	*Pjoin, Brjoin*1	✗	cross-product	✓
SPARQL Hybrid RDD	✓	*Pjoin, Brjoin+*	✓	cost-based	✗
SPARQL Hybrid DF	✓	*Pjoin, Brjoin+*	✓	cost-based	✓

Fig. 3.1 Quanlitative analysis of 5 query processing methods [1]

3.3 Vertex Centric Approach

In addition to above, there are some specialized systems for large graph processing, including Pregel [15] , Giraph [Giraph] , GraphLab [28], and GraphX [27]. These systems use vertex centric computing approach, in which a user defined function is executed in each node, and messages are send across edges and until all messages has not been received the computations is held standby . According to my search Eric L. Goodman seems to be the first to use the vertex centric framework of GraphLab for evaluating SPARQL queries [10]. Following Goodman there are two more works on GraphX. One is SPARQL over GraphX by Besat Kassaie [14]. His approach is mainly centered on subgraph matching , and aggregateMessages in GraphX have been used. The other work on GraphX is Spar(k)ql [8] . they also have used message passing but, a messge plan has been generated to reduce the number

of messages send between the nodes. Our work is similar to [11], and we have used their code and method for parsing the rdf data into RDDS, and then we have created DF.

3.4 Advantages of DataFrames over RDDS

DataFrame API which was introduced in version 1.3 of Saprk, but RDDS were with spark from the beginning. If we search, the work of Bagmeet Behera is Senior Data Scientist in the industry [1] on comparing the performance of RDDS and DF is a noticeable one. In his experiment , he has used a distributed running spark to analyze Geo-events data. On five different test cases, DataFrame and RDDS has been used for 5 different tasks, and the running time is noted down. His observation suggest that there is a significant different in the performance between the DF and the RDD for all the cases. Table2.1. RDD which is known for being fast, is doing good only on 2 of the 5 cases, and other 3 cases DataFrame outperforms the RDD as can be seen from the table. More interestingly is in RDD map() makes a tupel fo two atommic fields. In DataFrames, a GroupBy on 2 fields seems very fast in comparison [1] . The experiment concludes that the DF is faster that RDD for simple grouping and aggregations, but when there is new columns to be generated RDD when as in case 2. It shows that for very large Datasets the new DF API is indeed very fast.

TestCase	Aggregation
1	total events/hour
2	events count /hour
3	events count /publisher
4	events count /app/publisher

Table 3.1 5 differnt cases of the experiment (Source [1])

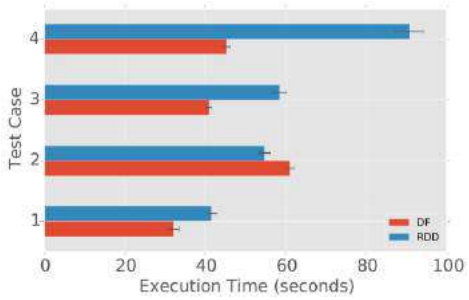


Fig. 3.2 Run-times of DataFrame(DF) and the Resilient-Distributed-Data(RDD) [1].

Chapter 4

Methodology

To answer SPARQL queries on a large distributed data,we use motif finding capability of GraphFrames. Graphframes gives us the efficiency of dataframes and cluster computing framework of spark . Like spar(k)ql [8] ,we have used LUBM data generator to generate the data, and Then the generated data,which is in .owl format, is converted to ntriple format. In this format, data is arranged in sequences of dot separated triples(subject-predicate-Object) and can easily be parsed by scala or any other programming languages.

4.1 Converting RDF to GraphFrames

To create the GraphFrames graph object, we need two DataFrames, one is the vertices and the other is the edges. Vertices can have as many column as it needs, but edges have only three columns, source , destination and label. Source and destination are the unique id in the source and destination vertices respectively and indicates an edge from source to destination Ids labeled by the label column.

4.1.1 Vertices and Edges

RDF triples are consisting of subject-predicate-object. Reading the input file line by line, Whenever the object of a triple is another resource , it is considered as an edge, and when the object is a literal, the literal is considered as the property of the nodes. Though rdf: type labels is used in RDF too relate the subject to another Object which is not a literal/property, treat it as a property and store them in vertices DF as a property of the corresponding subject, Like in the in spar(k)ql on GraphX, that reduces the graph size as well . Since a subject can be in many different lines of the ntriple file , each time to relate the subject with a different

literal or resource , each subject is mapped to a unique id, and at the end nodes with the same ids are combined. Bellow is sample of the DataFrames created.

ID	Vertice
18624	18624 undergraduateStudent351 type:Person telephone: xxxxxxx
57436	57436 undergraduateStudent195 type:Person department: department0
32196	32196 graduateStudent39 type:Person type:student researchinterest: researchgroup2
54040	54040 vourse51 type:graduatecourse name: course51
59768	59768 researchgroup4 type:researchgroup

Table 4.1 Vertice DataFrame

source	destination	label
77944	52863	takesCourse
31039	42688	takesCourse
59720	75819	publicationAuthor
14920	32072	undergraduateDegreeFrom
52948	78621	advisor

Table 4.2 Vertice DataFrame

Our approach is based on motif finding in GraphFrames. A Motif is defined as the structural patterns in the graph. A domain specific language (DSL) is used for representing motifs. In this DSL a node is represented as some sequence of one or more characters inside brackets e.g (X). An edge is represented as some sequence of one or more characters inside square brackets e.g [e1]. Unit of motif is a node-edge-node separated by "->" ((a)->[e1]->(b)), and each unit is separated by a semicolon. For example a motif of the form "(x)->[e1]->(y); (y)->(e2)->(x)" stands for all two nodes x and y that have links from x to y and from y to x. using this DSL ,motif finding in graph will return a DataFrame of all such structures in the graph, with columns for each of vertices and edges in the motif. In this case, the returned columns will be "x, y, e1, e2." When the motif DataFrame is returned, we can filter it column wise, based on the constants given in the SPARQL query and get all the matched rows.

4.2 Filtering The Motif

In order to filter the motif DataFrame, we ranked the edges based on the frequency of the edges in the graph. Lower the frequency higher the rank. The reason for this ranking is if we

filter based on a condition that returns lesser number of rows, the next condition will have to check lesser number of rows for further filtering the data.

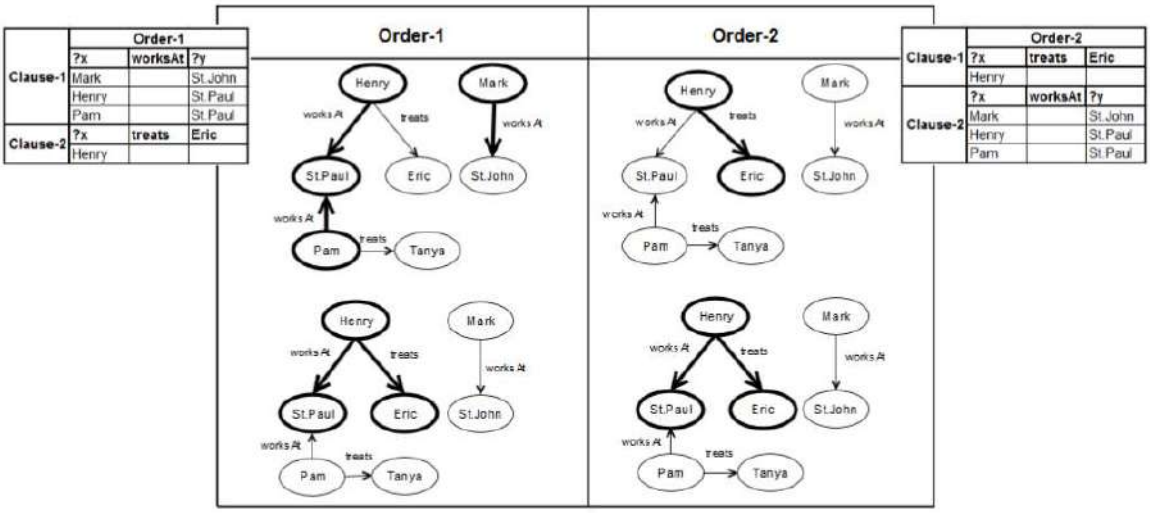


Fig. 4.1 The effects of ranking and query order

Table 4.1 models the motif coresponding to filtering without ranking and table4.2 represents the motif coresponding to filtering with ranking of the graph shown in the fiqure 4.1 . You can observe that in case where there is no ranking, we still have to filter 3 rows, while in case of ranked filtering there is only one row to be checked. A complete flow of our code is shown (see (Fig. 4.2)).

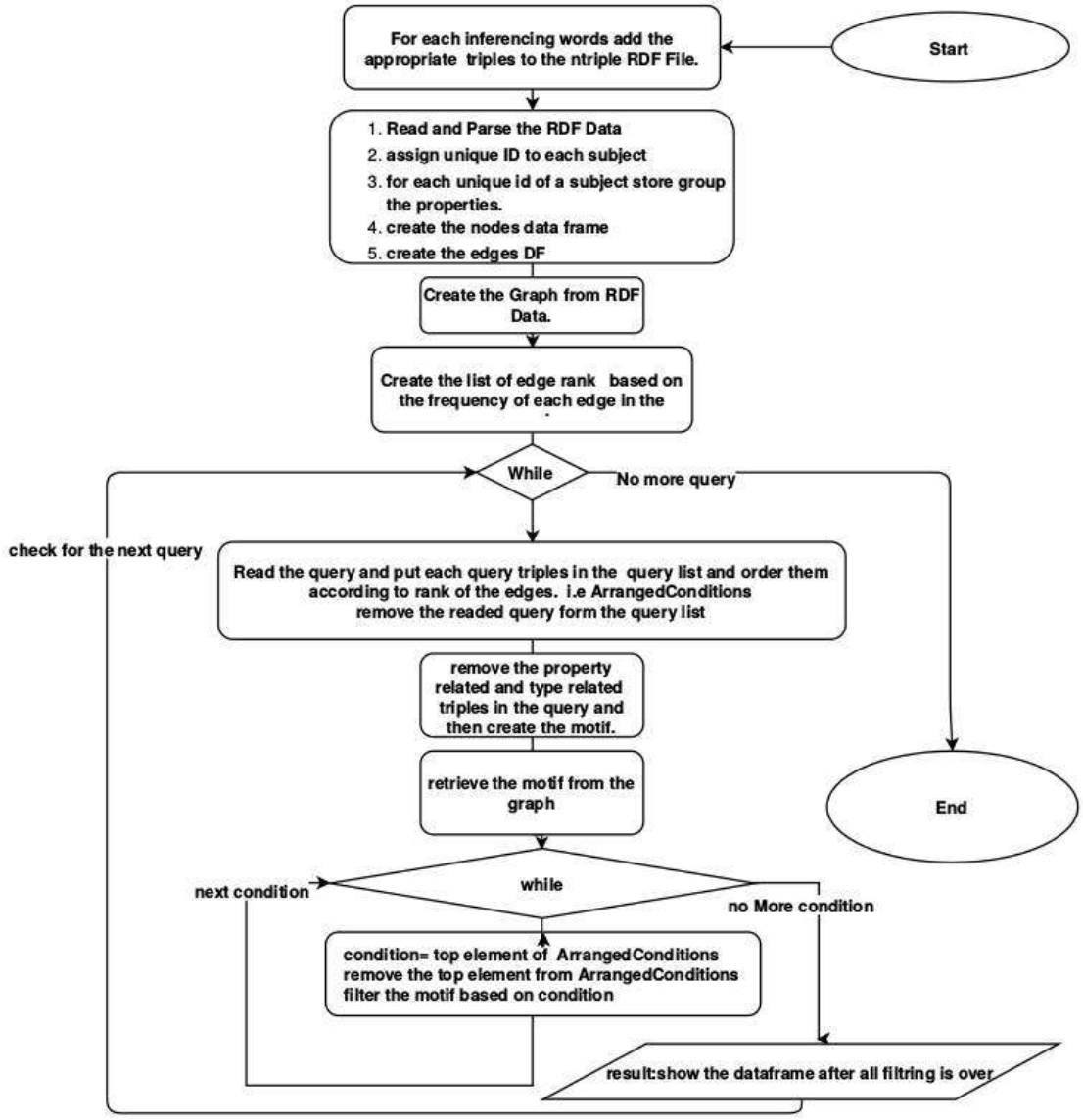


Fig. 4.2 Flow of the program

Chapter 5

Performance Evaluation

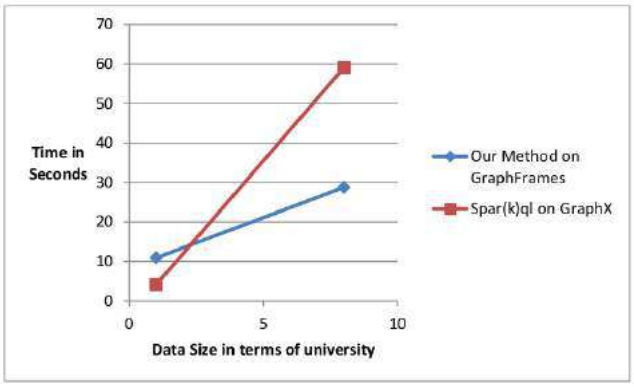
The result of our experiments with both the systems suggest that SPARQL by motif finding in GraphFrames is more efficient than message passing in graphX. we have tested both approaches for data generated for 1, 2, 3 and 8 universities respectively. In our system for smaller data sizes, we could implement the 12 of the 14 queries with GraphFrames approach. query no 13, we did not implement because of the inferencing rule which we have not implemented for inverse rule, and query no 14 requires no motif, and only needs linear search through vertices. the spar(k)ql approach on our system could be implemented for 6 queries only in case of data generated for 1 university (table. 5.1)and 4 queries in case of data generated for 8 university. (table. 5.2), the rest of the queries caused error of RpcTimeout, out of memory: java heap space, and time out exception as shown in the two tables for the result . Since only 4 queries could be run both for data generated for 1 university and 8 university in case of spar(k)ql , we have a comparison graph for these 4 queries .(Fig. 5.1)As it can be observed from the figure the rate at which running time of a query grows, is much larger in case of spar(k)ql in comparison with the motif finding approach in GraphFrames. Queries that we have not tested, NA is written in the corresponding cell .

Query	Duration in seconds	
	Message Passing in GraphX	Motif Finding in GraphFrame
Q1	4.097685737	10.907646878
Q2	JHS	21.689016381
Q3	3.770053958	7.817498369
Q4	5.456659175	13.943774281
Q5	3.934893670	8.558685320
Q6	2.922531009	NA
Q7	JHS	22.170295517
Q8	JHS	15.871218018
Q9	JHS	21.514006647
Q10	4.403382067	8.423265058
Q11	JHS	7.227994608
Q12	JHS	11.077609447
Q13	NA	NA
Q14	3.387621889	NA

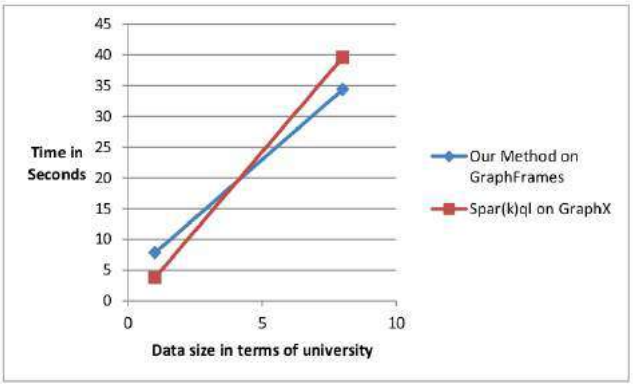
Table 5.1 Time Taken By Queries one University Data
(JHS :java heap space error, NA:not available)

Query	Duration in seconds	
	Message Passing in GraphX	Motif Finding in GraphFrame
Q1	59.114818420	28.760630212
Q2	JHS	21.689016381
Q3	39.636640871	34.393798319
Q4	172.604471974	47.328392276
Q5	36.181383695	43.185600770
Q6	RpcTimeout	NA
Q7	RpcTimeout	356.753187898
Q8	RpcTimeout	133.893332923
Q9	JHS	177.816715774
Q10	JHS	80.688538700
Q11	Time out exception	80.002803679
Q12	time out exception	98.758427080
Q13	NA	NA
Q14	JHS	NA

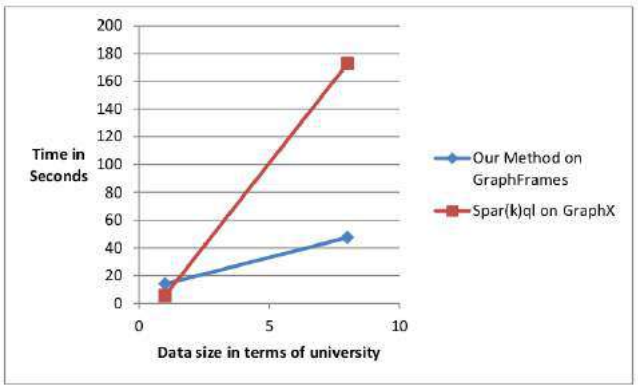
Table 5.2 Time Taken By Queries Eight University Data
(JHS:java heap space error, NA:not available)



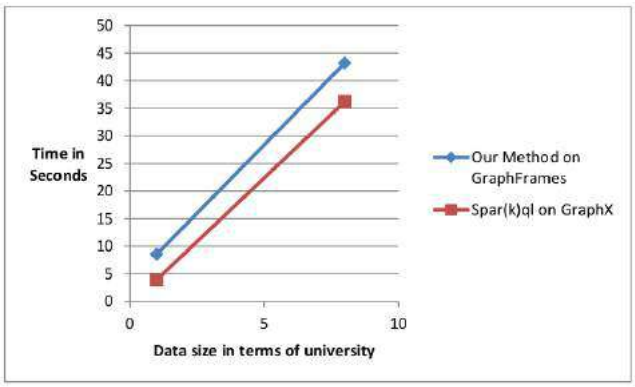
(a) Query1



(b) Query3



(c) Query4



(d) Query5

Fig. 5.1 Execution time with vs data size in spar(k)ql and motif finding approach in GraphFrames

Chapter 6

Conclusion

There is not much work on evaluating SPARQL queries on a distributed platforms, and it is new research area. In this research we used the distributed graph processing platform , GraphFrames for querying RDF data. Indeed we are the first to use graph for SPARQL since its release in 2016. In our work we could answer 13 of the 14 standard Lubm bench mark queries,for different data sizes. We tested these queries on GraphFrames ,on a standalone setup of spark and on a system with 16 GB of RAM only. We also compared our work with the previous work on graphX [27] that performs better than s2x [7] . We observed that for data size generated for one university we could answer 5 of the 14 queries, remaining giving error due limited memory space, while for our method on GraphFrames we could answer 13 of the 14 queries with out any error . Interesting though was that for the small data size of 1 university those 6 queries that could be run with error on graphX had a lower run time on graphX with spar(k)ql method than with our method on GraphFrames , but as the data size increased more of the queries hit error with previous method on graphX , and only 4 of the queries could be run without error for data generated for one eight university, but this with the response time in some cases much larger than our method on GraphFrames. Infact we observed that in one hand motif finding approach on GraphFrames for answering SPARQL queries is more reliable as there was no error of any type due to limited memory space /time out or any other type of error, and in the other hand we observed the efficiency of DataFrames, as our approach was based entirely on DataFrames.

References

[1] ad (2016). Comparing performance of spark dataframes api to spark rdd. [online] http://www.adsquare.com/comparing-performance-of-spark-dataframes-api-to-spark-rdd/#prettyPhoto.

[2] Alexander Schätzle, Martin Przyjaciel-Zablocki, G. L. (2014). Pigsparql: Mapping sparql to pig latin. *Proceeding SWIM '11 Proceedings of the International Workshop on Semantic Web Information Management*, page 167 – 179.

[3] Ankur Dave, Alekh Jindal, L. E. L. . X. J. G. M. Z. (2010). Graphframes: An integrated api for mixing graph and relational queries. *Spark Summit East 2016,Data Science and Engineering at Scale,February 16 – 18 New York*, page 265–276.

[4] Association, I. R. M. (2016). *The TEXBig Data: Concepts, Methodologies, Tools, and Applications*. IGI Global.

[5] DaveBeckett (2011). What does sparql stand for?

[6] EricPrud (2008). Sparql query language for rdf. [online] http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115.

[7] et al, A. S. (2015). S2x: Graph-parallel query- ing of rdf with graphx. *In: Proc. of 1st International Workshop on Big-Graphs Online Querying*.

[8] Gergő Gombos, Attila Kiss, G. R. (2017). Spar(k)ql: Sparql evaluation method on spark graphx. *Future Internet of Things and Cloud Workshops (FiCloudW), IEEE International Conference on*.

[Giraph] Giraph. Apache giraph. [online] http://giraph.apache.org/.

[10] Goodman, E. L. and Grunwald (2014). Using vertex-centric programming platforms to implement sparql queries on large graphs. *Fourth Workshop on Irregular Applications: Architectures and Algorithms (Piscataway, NJ, USA, 2014), IA3 '14, IEEE Press*, page 25–32.

[11] Hubert Naacke, Bernd Amann, O. C. (2016). Sparql query processing with apache spark. *semanticscholar*.

[12] JimRapoza (2007). Sparql will make the web shine. [online] semantic-web@w3.org.

[JSON] JSON. Json-ld 1.0: A json-based serialization for linked data.

[14] Kassaie, B. (2017). Sparql over graphx. *Cornel University Library*.

[15] Malewicz, G., A. M. H. B. A. J. D. J. C. H. I. L. N. and Czajkowski (2010). Pregel: A system for large-scale graph processing. *ACM SIGMOD International Conference on Management of Data*, pages 135–146.

[16] Mohammad Farhan Husain, Pankil Doshi, L. K. and Thuraisingham, B. (2014). Storage and retrieval of large rdf graph using hadoop and mapreduce. *Proceeding CloudCom '09 Proceedings of the 1st International Conference on Cloud Computing*, page 680 – 686.

[17] Nikolaos Papailiou, Ioannis Konstantinou, D. T. and Koziris, N. (2014). Adaptive query processing on rdf data in the cloud." computing systems laboratory. *School of ECE, National Technical University of Athens*, page 680 – 686.

[18] NQuads (2014). N-quads: Extending n-triples with context. [online] semantic-web@w3.org.

[19] RDF (2004). Resource description framework. [online] https://www.w3.org/RDF/.

[20] RDFNTriples (2014). Rdf 1.1 n-triples: A line-based syntax for an rdf. [online] semantic-web@w3.org.

[21] RDFTurtle (2014). Rdf 1.1 turtle:terse rdf triple language. [online] semantic-web@w3.org.

[22] RDFXML (2014). Rdf 1.1 xml syntax. [online] semantic-web@w3.org.

[23] ReynoldXin (2011). Introducing dataframes in apache spark for large scale data science. [online] https://databricks.com/blog/2015/02/17/introducing-dataframes-in-spark-for-large-scale-data-science.html.

[24] Segaran, Toby; Evans, C. T. J. (2009). *Programming the Semantic Web*. O'Reilly Media, 1005 Gravenstein Highway North, Sebastopol.

[25] spark2 (2011). Spark sql, dataframes and datasets guide. [online] http://spark.apache.org/docs/latest/sql-programming-guide.html.

[26] sparql (2004). sparql query. [online] https://www.w3.org/TR/rdf-sparql-query/.

[27] Xin, R. S., G. J. E. F. M. J. and Stoica (2013). Graphx: A resilient distributed graph system on spark. *First International Workshop on Graph Data Management Experiences and Systems, ACM*.

[28] Zeng, K., Y. J. W. H. S. B. and Wang (2010). A distributed graph engine for web scale rdf data. *the 39th international conference on Very Large Data Bases (Trento, Italy, 2013)*, page 265–276.

Appendix A

List of the Queries used

The 14 Standard LUBM Bench mark Queries used with the coresponding motifs:

Query1
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://www.lehigh.edu/ zhp2/2004/0401/univ-bench.owl#>
SELECT ?X
WHERE
{?X rdf:type ub:GraduateStudent .
?X ub:takesCourse
http://www.Department0.University0.edu/GraduateCourse0}
(X)-[no]->(da)
Query2
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://www.lehigh.edu/ zhp2/2004/0401/univ-bench.owl#>
SELECT ?X, ?Y, ?Z
WHERE
{?X rdf:type ub:GraduateStudent .
?Y rdf:type ub:University .
?Z rdf:type ub:Department .
?X ub:memberOf ?Z .
?Z ub:subOrganizationOf ?Y .
?X ub:undergraduateDegreeFrom ?Y}
(X)-[jh]->(Z); (Z)-[ci]->(Y); (X)-[yx]->(Y)

```
# Query3
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://www.lehigh.edu/ zhp2/2004/0401/univ-bench.owl#>
SELECT ?X
WHERE
{?X rdf:type ub:Publication .
?X ub:publicationAuthor
http://www.Department0.University0.edu/AssistantProfessor0}
(X)-[as]->(xs)
# Query4
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://www.lehigh.edu/ zhp2/2004/0401/univ-bench.owl#>
SELECT ?X, ?Y1, ?Y2, ?Y3
WHERE
{?X rdf:type ub:Professor .
?X ub:worksFor <http://www.Department0.University0.edu> .
?X ub:name ?Y1 .
?X ub:emailAddress ?Y2 .
?X ub:telephone ?Y3}
(X)-[ge]->(ok)
# Query5
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://www.lehigh.edu/ zhp2/2004/0401/univ-bench.owl#>
SELECT ?X
WHERE
{?X rdf:type ub:Person .
?X ub:memberOf <http://www.Department0.University0.edu>}
(X)-[mz]->(xn)
# Query6
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://www.lehigh.edu/ zhp2/2004/0401/univ-bench.owl#>
SELECT ?X WHERE {?X rdf:type ub:Student}
# Query7
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://www.lehigh.edu/ zhp2/2004/0401/univ-bench.owl#>
SELECT ?X, ?Y
```

```
WHERE
{?X rdf:type ub:Student .
?Y rdf:type ub:Course .
?X ub:takesCourse ?Y .
<http://www.Department0.University0.edu/AssociateProfessor0>,
ub:teacherOf, ?Y}
```
(X)-[dk]->(Y); (fy)-[lp]->(Y)
```
# Query8
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://www.lehigh.edu/ zhp2/2004/0401/univ-bench.owl#>
SELECT ?X, ?Y, ?Z
WHERE
{?X rdf:type ub:Student .
?Y rdf:type ub:Department .
?X ub:memberOf ?Y .
?Y ub:subOrganizationOf <http://www.University0.edu> .
?X ub:emailAddress ?Z}
```
(X)-[ds]->(Y); (Y)-[ba]->(vd)
```
# Query9
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://www.lehigh.edu/ zhp2/2004/0401/univ-bench.owl#>
SELECT ?X, ?Y, ?Z
WHERE
{?X rdf:type ub:Student .
?Y rdf:type ub:Faculty .
?Z rdf:type ub:Course .
?X ub:advisor ?Y .
?Y ub:teacherOf ?Z .
?X ub:takesCourse ?Z}
```
(X)-[xa]->(Y); (Y)-[im]->(Z); (X)-[ju]->(Z)
```
# Query10
# to the results.
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://www.lehigh.edu/ zhp2/2004/0401/univ-bench.owl#>
SELECT ?X
WHERE
```

{?X rdf:type ub:Student .
?X ub:takesCourse
<http://www.Department0.University0.edu/GraduateCourse0>}
(X)-[oz]->(cw)
Query11
Additionally, its input is small.
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://www.lehigh.edu/ zhp2/2004/0401/univ-bench.owl#>
SELECT ?X
WHERE
{?X rdf:type ub:ResearchGroup .
?X ub:subOrganizationOf <http://www.University0.edu>}
(X)-[fw]->(xq)
Query12
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://www.lehigh.edu/ zhp2/2004/0401/univ-bench.owl#>
SELECT ?X, ?Y
WHERE
{?X rdf:type ub:Chair .
?Y rdf:type ub:Department .
?X ub:worksFor ?Y .
?Y ub:subOrganizationOf <http://www.University0.edu>}
(Y)-[vs]->(wu)
Query13
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://www.lehigh.edu/ zhp2/2004/0401/univ-bench.owl#>
SELECT ?X
WHERE
{?X rdf:type ub:Person .
<http://www.University0.edu> ub:hasAlumnus ?X}
Query14
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://www.lehigh.edu/ zhp2/2004/0401/univ-bench.owl#>
SELECT ?X
WHERE {?X rdf:type ub:UndergraduateStudent}