SimbaQL: A Query Language for Multi-source Heterogeneous Data

Yuepeng Li[1,2], Zhihong Shen[1], and Jianhui Li[1(✉)]

[1] Computer Network Information Center, Chinese Academy of Sciences, Beijing 100190, China
{liyuepeng,lijh}@cnic.cn
[2] University of Chinese Academy of Sciences, Beijing 100049, China

Abstract. In a data-driven era, scientific discovery by querying integrated heterogeneous data is becoming a popular approach. However, most current search engines retrieve data using SQL, which only addresses part of the common data processing use cases in data discovery for scientific research. Besides, the expressive power of SQL and other popular languages can't describe some simple but useful second order logic queries. Therefore, we first proposed an abstract data processing model which contains four components: data set, data source, data model, and analysis tool. Then, we introduced a unified data model and SimbaQL query language to describe the steps of data processing. At last, we studied two cases by describing the data processing using SimbaQL, and it turns out that SimbaQL can describe these tasks properly.

Keywords: Query language · Heterogeneous data · Scientific data

1 Introduction

Since the concept of "Big Data" was proposed in the 2011 Mckinsey annual report, the definition of "Big Data" has been in dispute on academia and industry [1]. Most of the dispute came from the characteristics of big data in different domains. And in scientific research, data sharing is an obvious different characteristic with other domains [2]. For example, most of nowadays significant scientific programs, such as Human Genome Project (HGP), Large Hardon Collider (LHC), etc. share data around the world. And scientists make discoveries from the open data, including the discoveries of gravitational wave and god particle which win the Nobel prize. Generally, several open source data will be involved in a scientific research. For example, the World Data Center for Microorgannisms (WDCM) [3] published a knowledge graph that is built from 36 open source data including Taxonomy, Genbank, Gen, etc. These data has various formats, such as file, relational database etc. BigDAWG [4] is a polystore database, which manages the medical record, sequence data, image, radio etc. for medical research. The applications mentioned above have a similar characteristic that all of them

J. Li et al. (Eds.): BigSDM 2018, LNCS 11473, pp. 275–284, 2019.
https://doi.org/10.1007/978-3-030-28061-1_27

need search heterogeneous data in data processing. However, the query language of current tools for heterogeneous data cannot describe the whole life cycle of these data processing. Therefore, we proposed a query language for multi-source heterogeneous data, which can import/export data to database, transform data from different data models and integrate multi-source data.

2 Related Work

Multiple types of databases have been used to manage data in nowadays system, such as relational database, graph database, document database, key-value database and so on. These databases build data models to describe the data formally, and retrieval data using the operations or query language of data model. For example, relational databases query the relational data model by SQL; graph databases describe data by property graph model and RDF, and query the graph by Cypher or SparQL; Xquery language search object model for XML file [5].

When it comes to heterogeneous data, there are three methods to access data: data integration, multi-model database, polystore [4]. Data integration [6] is defined as a three tuples (G, L, M), where G is global data model, L is local data model, and M is the mapping from local data models to global data model. Data integration queries local data models using global data model query language. For example, popular-used SQL query engines, such as Impala, SparkSQL, Presto, query multiple data sources using SQL language. Besides, data warehouse, federated database [7] are also methods of data integration. Multi-model database [8] stores different types of data using one unified data model, and query data using unified language, such as OrientDB [8], Ag- groDB, etc. Polystore database stores data in different databases. When query by language A, polyestore database execute the query in A database. If the target data is not stored in A database, then polyestore will CAST the target data to A database [9].

However, above methods for heterogeneous data have several defects. First, data integration mainly use SQL as global query language, but NoSQL databases are de- signed to against SQL, therefore SQL is obviously not suitable for querying different data model. Second, multi-model databases store data in one backend, which is not suitable for multi-source data. Besides, although polystore databases can query data in different ways, but it require users to learn many database languages, which means polystore just transform data between different data models and cannot achieve the goal of unified search. And current polyestore query language only supports one time transform, and cannot query the intermediate results many times.

As regard as the query language for heterogeneous data, current query language have another two defects. On one side, current query language is not designed for data integration specially. For example, in the processing of build knowledge graph, WDCM wants to create the relation between species's information stored in relational database and the gene sequence information stored in Genbank file. However, SQL failed to treat species and genbank file as a

whole to extract and integrate data. On the other side, current languages are good at dealing with different type of data, and most of them can only solve first logic problems [10–12]. For example, relational data model describe relationships between objects using foreign key, which means we can only get the relationship between objects by computing. As for Cypher [13], it can only query related data stored in database. The following query is illegal

```
match (p:Person), (p2:Person)
where p.name=p2.name
```

To solve the problems mentioned above, we first proposed an intermediate data model which can transform between common data models. Then, we defined the operations and query language on the data model. At last, we studied the query language by two scientific cases. The main contributions of this paper are as followed. First, we abstract the scientific data processing by data set, data source, data model, analyzing tools. Second, we designed data model, Linked-Document-Model, and query language, SimbaQL, for data integration specialty. Third, we enhance the expressive power by introducing the operation of describing the property of LDM.

3 Methodology

In big data era, scientific researches are mainly driven by data. If one scientist wants to find which factor is the most important to a country's happiness, he need to get the published GDP data, agriculture data, political data and so on. These data may be located in files, web service, database and so on. Then, he needs to import theses data into databases, and extract interested part from database. Finally he will integrate all the data to the format that analyze tools can recognize.

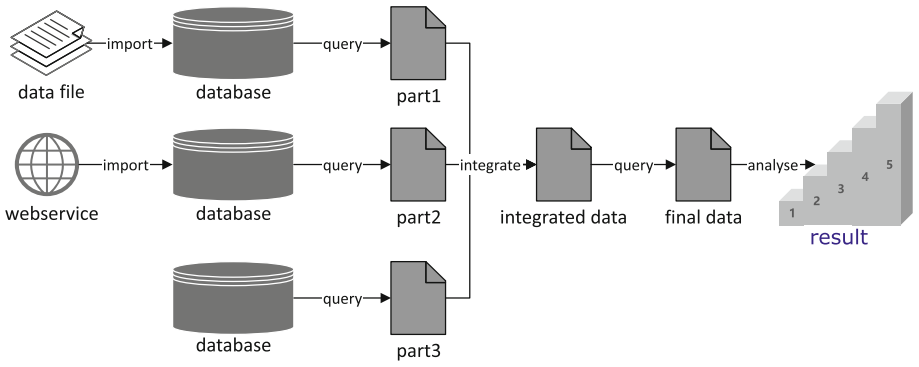

Fig. 1. Data processing in scientific research.

Above data processing is not alone. As Fig. 1 shows, the last step in scientific discoveries is analyzing data. However, most of scientist's work are preparing

data from heterogeneous sources. In the process of preparing data, three types of components are involve: *data set, data source and analyzing tools*. Data set, such as file, can only store data. Data source, such as database, retrieval engine, can not only store data, but also manage data. Analyzing tools, such as Spark, can only process data. We can divide the data process into the following steps:

1. *Find open source data set or build his own data set.*
2. *Import data set into proper data source.*
3. *Search interested data and integrate them.*
4. *Reprocess the integrated data to the format that analyzing tools can recognize.*
5. *Analyze the data by tools.*

In this paper, we want to design a query language that can accomplish the tasks in 1- 4 steps. For the first step, we define different data set types, such as file, http, web- service, to get the target data. For the second step, we define the import operation of data set. For 3–4 steps, we defined intermediate data model LDM and its mapping rules and operations. Besides, we define the description operation to enhance the expressive power of SimbaQL, and export operation to push the LDM data to analyzing tools. Our abstract model of data processing in scientific research is shown as Fig. 2.

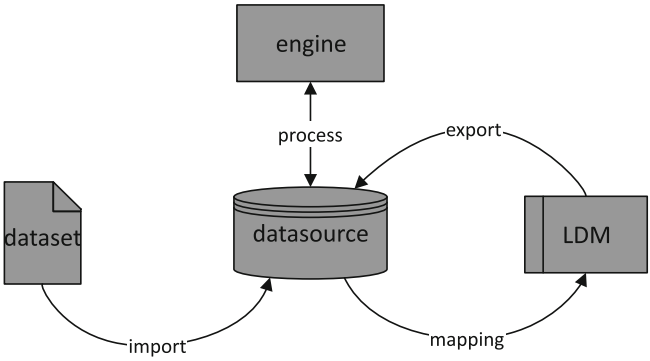

Fig. 2. Data processing in scientific research.

As Fig. 3 shows, heterogeneous data query system that implement SimbaQL contains three components: SimbaQL parser, execute engine and data source. When user input a query, the parser will translate it into data set, data source, and operations of LDM. For example, if we want to query a csv file, the system may import the file data set to relational database, and then perform the query. The system will invoke the query en- gine of this data source when a query only involve a single data source. And If one or many data sources cannot solve the query problem, then execute engine will make a compensate computing for this query.

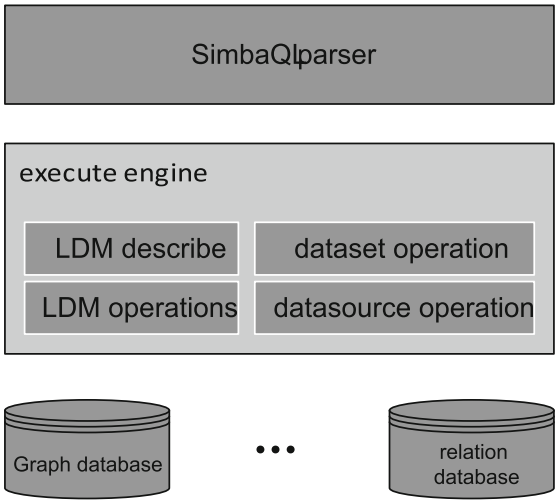

Fig. 3. Data processing in scientific research.

4 Simba Query Language

In this section, we proposed a unified data model LDM, which contains four types of operations. After that we defined the mapping rules between four often-used data mod- els and LDM. At last, we introduced the informal SimbaQL language grammar.

Linked-Document-Model. The goal of data model is to describe the objective things and relationship between things formally. There are two methods to describe a thing: schema and schemaless. Schema is more suitable for computing, but schemaless way is more suitable for altering thing dynamically. Similarly, there are two methods to describe the relationships between things: storage and computing. Relational data model use computing to get relationship, but the relationship has less semantic. Graph model use storage to get relationship, but it costs storage space.

To satisfy unified query for heterogeneous data, we want LDM has the following three characteristics.

1. *Support both schema and schemaless ways to describe things.*
2. *Support both storage and computing ways to describe relationship.*
3. *Support describing the data processing in scientific research.*

Link and Document. Document is a semi-structured document which has multiple attributes. Each document must has unique key to identify this document. We can add attributes to a document dynamically, and we can also require

a document must have some attributes. Generally, we are talking about document set which must have a label name, and one document can belong to multiple document sets. Link is a special document which must have two attributes *(from, to)*. Similar with document set, link set must have a label name. We can define a link set by two ways. For example, we can define the "knows" link by storage way,

$$knows = [(from : 1, to : 2), (from : 2, to : 3)]$$

and we can also define the link by computing way

$$knows = \{(from, to) | from \in customer.id \quad and \quad to \in order.customerid \quad and \quad customer.id = order.customerid\}$$

Mapping Rule. LDM is a tuple defined as followed. The first item of this tuple is

$$ldm = (\{document\ sets\}, \{link\ sets\})$$

a set of document set, and the second item is a set of link set. A LDM can stand for a data source where the document set and link set in LDM is mapped to the components of data source. For example, table in relational database is mapped to a document set, and foreign key is mapped to a link set. As Table 1 shows, we define the mapping rules between relational data model, graph data model, key-value data model, document data model and LDM.

Table 1. LDM mapping rules.

LDM	Relational model	Key-value mode	Document model	Graph model
attribute	attribute	key	attribute	attribute
document	record	pair	collection	vertex
document set	table	——	——	——
link	——	——	——	edge
link set	foreign key	——	——	——

Operations. There are four types of operations on LDM: set operation, link set operation, document set operation and transform operation. The detail of these operations are shown as Table 2. Besides, we can access the data in LDM similar to url, the url is defined as followed.

$$< datasource > . < document > . < links > . < index > . < attributes >$$

For example, wdcm.species.gen stands for the gene of the species.

Table 2. LDM mapping rules.

Operation	Description
Set Operation	
+(Union)	Union two LDM into one LDM. Combine the documents and Links with same label
∩(Intersect)	Find the same documents and links between two LDMS
−(Minus)	Find the documents and links which are in LDM1 and not in LDM2
Link Set Operation	
×(Create Link)	Create links between documents based on given conditions
δ_{link}(Projection)	Project a LDM into a link means filter the data in document which exist some link
π_{link}(Selection)	Select the links which has the feature such as name, count, etc
Document Set Operation	
δ_{atr}(Projection)	Aimed at attributes which have some features. If the documents does not have an attribute, the document is not in the result LDM
π_{atr}(Selection)	Aimed at attributes which have some features. If the attribute of a document does not satisfy the condition, the document is filtered
Transform Operation	
todocs	Transform the LDM to different type of documents
tograph	ransform a LDM into a graph
totables	Transform a LDM into different relation tables, including the Link

Description. We use second order logic to describe what characteristic a LDM should have. And the way we describe characteristic is by rules. For example, we first define a social network, which has *person* document set and *knows* link set.

$$soc = (\{person\}, \{knows\})$$

We can require the person set satisfy a transitive closure characteristic: suppose that *p1,p2,p3* are documents in person, if *p1* knows *p2*, and *p2* knows *p3*, then *p1* must knows *p3*. This closure can be described as followed,

```
soc.person p1,p2,p3;
p1.knows=p2 and p2.knows=p3 -> p1.knows=p3;
p1,p2,p3 >>
```

The last line stands for which part of the set that *soc.person* should contain. Another case is the person set stratify the condition that it contains all the person that "simba" indirectly knows. The characteristic is defined as followed

```
define x.indirknows=y: x.knows.knows=y or
x.indirknows=z and z.knows=y;
soc.person p1,p2;
p1.name= ""simba;
p2->p1.indirknows=p2;
p2>>
```

The keyword 'define' defines a new link named 'indirknows'.

4.1 SimbaQL Grammar

As described in Sect. 3, SimbaQL need to describe the whole data processing in scientific research. We divide the grammars into four parts: data set grammar, data source grammar, LDM operation grammar, and LDM description grammar. We use keyword *'create'* to define a data set or data source, and a data set can call the *'importto'* function to import a data set to data source. Data source can call *'query'* function to search the local database. At the same time, data source is also a LDM, thus can perform the operations of LDM. For example,

```
create genfile = File(*);
genbank = genfile.importto'('mongodb);
a_class_gen = genbank.query({name: "a_class"});
```

The most used three operations in LDM are selection, projection, union. The grammars for theses operations are as follows,

```
a_class_gen = genbank.gen(name= "a_class");
b_class_gen = genbank.gen(name= "b_class");
ab_class_gen = a_class_gen + b_class_gen;
ab_class_gen = ab_class_gen[name, sequence];
```

For the LDM description, we use keyword *'st'* to represent this LDM should satisfy the following characteristic. For example

```
newsoc = soc st {
  soc.person p1,p2;
  p2->p1.knows=p2;
  p2>>
}
```

5 Case Study

In this section, we apply SimbaQL in two actual scientific research cases. The first case is the data processing when building the WDCM knowledge graph.

The knowledge graph contains the information of microorganism species, protein, gene and the relationship between them. In this process, WDCM creates a relational database to store the species's information, and they want to build the relationship between species and their gene sequence which can get from the file of genbank. This process can be implemented by SimbaQL as follows:

```
create species=JDBC(*);
create genbank=File(*);
genbank=genbank.import('mongo);
genbank=genbank.gen[sorcename,sequence];
wdcm=species+genbank;
wdcm.spece*wdcm.gen{wdcm.spece.name
.contains(wdcm.gen.name)};
wdcm.importto('gstore);
```

The second case is that National Nature Science Foundation of China has foundation information stored in relational database, and they want to know the mapping between paper and foundation. However, the papers published by a foundation are listed in the summary report. Therefore, they need to extract the papers from the reports, and integrated the data. The paper can be extracted by regular expression, and this process can be described by SimbaQL as follows:

```
create found=JDBC(*);
create reports=File(*);
reports=reports.importto('path_search);
papers=reports.query(paperegx);
found_paper=found+papers;
found_paper.found*found_paper.paper{found_paper.
found.name=found_paper.paper.filename};
found_paper.import('gstore);
```

6 Conclusion and Future Work

We introduced SimbaQL for multi-source heterogeneous data query which is a common data processing in scientific research. First, we propose an abstract model of data processing, which consists of data set, data source, data model, and analyzing tools. Then, we defined an intimidate data model LDM)to map the data between heterogeneous data, and operations on LDM to query and integrate data. We study two actual cases using SimbaQL, and it turns out that SimbaQL could describe the common data processing in scientific research. In the future, we will give the formal definition of SimbaSQL, prove the expressive power of language, and implement the query engine for heterogeneous data.

References

1. Ward, J.S., Barker, A.: Undefined by data: a survey of big data definitions, arXiv preprint arXiv:1309.5821 (2013)
2. Lu, J., Holubová, I., et al.: Multi-model data management: what's new and what's next? (2017)
3. World data centre for microorgannisms. http://www.wdcm.org/
4. Duggan, J., et al.: The BigDAWG polystore system. ACM Sigmod Rec. **44**(2), 11–16 (2015)
5. Chamberlin, D.: XQuery: an xml query language. IBM Syst. J. **41**(4), 597–615 (2002)
6. Lenzerini, M.: Data integration: a theoretical perspective. In: Proceedings of the Twenty-First ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, pp. 233–246. ACM (2002)
7. Sheth, A.P., Larson, J.A.: Federated database systems for managing distributed, heterogeneous, and autonomous databases. ACM Comput. Surv. (CSUR) **22**(3), 183–236 (1990)
8. Baranyi, P.: Multi-model database orientdb. https://orientdb.com/
9. Gadepally, V., et al.: Version 0.1 of the BigDAWG polystore system, arXiv preprint arXiv:1707.00721 (2017)
10. Libkin, L.: Expressive power of SQL. Theor. Comput. Sci. **296**(3), 379–404 (2003)
11. Angles, R., Gutierrez, C.: The expressive power of SPARQL. In: Sheth, A., et al. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 114–129. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-88564-1_8
12. Bagan, G., Bonifati, A., Ciucanu, R., Fletcher, G.H., Lemay, A., Advokaat, N.: Generating flexible workloads for graph databases. Proc. VLDB Endow. **9**(13), 1457–1460 (2016)
13. Cypher-the neo4j query language. http://www.neo4j.org/learn/cypher