A Query Language of Data Provenance Based on Dependency View for Process Analysis

Xuan Sun
Beijing Information Science & Technology University
Peking, China
sunxuanbupt@126.com

Xin Gao
The National Computer network Emergency Response technical Team Coordination Center of China
Peking, China
gaoxin54@126.com

Huiying Du
Beijing Information Science & Technology University
Peking, China
huiyingdu@bistu.edu.cn

Wei Ye
National Engineering Research Center for Software Engineering, Peking University
Peking, China
wye@pku.edu.cn

***Abstract*—For the scale of data in process keep increasing, data provenance also becomes large and constantly growing, which brings challenges to the efficiency of provenance tracking in process analysis. This paper proposes a kind of dependency view to extract a global data provenance description of the data process instance, and then defines a contextual query language based on dependency view to implement an efficient provenance query mechanism for process analysis. The elements of the language are based on a set of dependency view query operations, which can decrease the steps of provenance tracking based on the elements of data provenance and support the descriptive power of the language for complex provenance tracking. Experimental results show that complex provenance tracking by the language is efficient and ease to use.**

Keywords—provenance; query language; dependency view

I. INTRODUCTION

Data provenance records the related operations and data in the execution of data process, which is regarded as an important data in many process-aware systems. In the application domains of data provenance, it is used for auditing in the data base and supporting analysis in the complex science experiment environment in the early times, process analysis and process verification following the development of workflow, semantic resolving in the web, and now existing as metadata in the cloud environment [1, 2, 3]. Following the development of the application of data provenance, we can find out that the scale of the data object supported by data provenance becomes more and more huge. Therefore it directly increases the amount of the intermediate data in the process of the source data. So now we need to satisfy the efficient requirement of data provenance analysis. Currently, many researchers try to deal with this problem based on distributed data management platform, like cloud platform, which takes the advantage of large-scale storage and computing power of cloud platform. Ikeda et al. [4] propose an approach for tracking the provenance of workflow modeled as MapReduce jobs. Malik et al. [5] introduce an approach for recording provenance in distributed environment and each node stores parts of entire provenance graph. Based on cloud environment, it can temporarily increase the efficiency of data provenance query, but its cost becomes higher when the scale of data provenance keep increasing and meanwhile it still needs to program the query function in the cloud environment. So we finally still need to optimize the query mechanism of data provenance when the query requirement becomes more and more complex as the data provenance dataset keeps growing. However, current data provenance models are always not ready for directly implement the query requirements, because they aim at describing the dependency relations among the elements of data provenance. So extracting the dependency relations from data provenance and providing higher view of data provenance is a way to improve the efficiency of data provenance query. In this paper, we focus on the efficient and usable querying of data provenance and carry out our research from the aspect of data provenance description and corresponding query mechanism. We try to extract dependency elements from data provenance directed graphs to form the dependency view and define corresponding query operations of dependency view for provenance tracking, which can make data provenance records suit to be queried. Then we propose a query language based on dependency view to describe the requirement of complex provenance tracking.

The remainder of this paper is organized as follows. We introduce the related work in section 2, introduce dependency view of data provenance in section 3, define the query operations for data provenance based on dependency view in section 4, propose the data provenance query language in section 5, and validate the efficiency of query language in section 6. In the end, we make our conclusions in section 7.

II. RELATED WORK

Data provenance as the key technology for data-intensive research provides a kind of causal relationship model among the result, operation, middle data, and human and so on. There has been significant progress on formal models for data provenance. However, the difficulty to support process analysis based on data provenance is mainly the complexity and variety of the analysis requirement, which proposes a serious description problem for data provenance querying. Then the issue of data provenance querying is just addressed in a application-independent way to in recent years, the query

DOI reference number: 10.18293/SEKE2016-049
This work was supported by the Scientific research project of Beijing Municipal Education Commission under Grant No.SM201511232004 and National Natural Science Foundation of China under Grant No.61202070.

languages of data provenance are proposed to resolve the problem, like OPQL [6], VQuel [7], ProQL [8], QLP [9], etc. These query languages are based on formal models for data provenance, which cause the difficulty in writing query by these languages. Therefore, we need a flexible and extensible query language of data provenance to support dependency deducing of data provenance while the complex and various requirements of process analysis are proposed.

III. DEPENDENCY VIEW OF DATA PROVENANCE

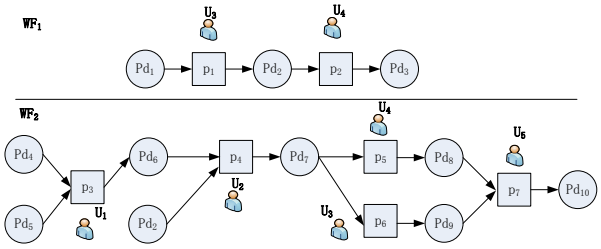

Figure 1. An execution instance of data process

For each task in the process execution, its dependency elements always include the related input data, tasks, operators and constraint, just as figure 1. All these dependencies consist of the context for the executing task, which can be extracted as directed graph from the executed part of the instance. We call these extracted directed graphs as dependency view of execution instance. According the type of the node in dependency view, we divide them into three categories: data dependency view, process dependency view and collaboration dependency view. The data dependency view is signed as **DFDepView**=<**D**, **DFDep**>. **D** stands for data set of input data and output data, and **DFDep** stands for the casual relationships of "was Derived From". The process dependency view is signed as **PFDepView**=<**P**,**PFDep**>. **P** stands for the task instance set, and **PFDep** stands for "was Triggered By". The collaboration dependency view is signed as **HFDepView**=<**H**, **HFDep**>. **H** stands for operator set, and **HFDep** stands for the collaboration relationships among **H**. So we use an abstract model <**N**, **E**> to describe these three dependency views, where **N** stands for the nodes in dependency view and **E** stands for the dependency relationship in dependency view. Assuming that **A** and **B** belong to the same kind of dependency view, and the operations for dependency view are defined as table I.

TABLE I. DESCRIPTION OF OPERATIONS FOR DEPENDENCY VIEW

Operation	Description
$\mathbf{A}\cup\mathbf{B}$=<$\mathbf{A}.\mathbf{N}\cup\mathbf{B}.\mathbf{N}$, $\mathbf{A}.\mathbf{E}\cup\mathbf{B}.\mathbf{E}$>	union operation for dependency view
$\mathbf{A}\cap\mathbf{B}$=<$\mathbf{A}.\mathbf{N}\cap\mathbf{B}.\mathbf{N}$, $\mathbf{A}.\mathbf{E}\cap\mathbf{B}.\mathbf{E}$>	intersection operation for dependency view
A-B=<**A.N-B.N**, {e|$e\in$ (**A.E-B.E**) $\wedge$ $e.source\in$ (**A.N-B.N**) $\wedge e.destination\in$(**A.N-B.N**)}>	complement operation for dependency view
$\mathbf{A}\oplus\mathbf{B}$=<$\mathbf{A}.\mathbf{N}\oplus\mathbf{B}.\mathbf{N}$, {$e$|$e\in$($\mathbf{A}.\mathbf{E}\oplus\mathbf{B}.\mathbf{E}$) $\wedge e.source\in$($\mathbf{A}.\mathbf{N}\oplus\mathbf{B}.\mathbf{N}$) $\wedge e.destination\in$($\mathbf{A}.\mathbf{N}\oplus\mathbf{B}.\mathbf{N}$)}>	symmetric difference operation for dependency view

Beside these basic operations, there are also the operations for the set of dependency view. Assuming the set **A** consists of $\mathbf{A}_1$, $\mathbf{A}_2$, …, $\mathbf{A}_n$ which belong to the same kind of dependency view, signed as $\mathbf{A}=\{\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_n\}$. Then the corresponding union and intersection operation can be carried out as following:

$$\cup\mathbf{A}=\mathbf{A}_1\cup\mathbf{A}_2\cup\ldots\cup\mathbf{A}_n;\ \cap\mathbf{A}=\mathbf{A}_1\cap\mathbf{A}_2\cap\ldots\cap\mathbf{A}_n$$

From the point view of dependency view, any OPM instance can transform into the three dependency views and can be analyzed based on the operations of dependency view. Thus dependency view provides a more coarse-grained description than OPM[10], which provides a base for the improvement of data provenance tracking. So we describe the context of the task at runtime based on dependency view in this paper, and try to deduce the requirement of provenance tracking based on dependency view. Assuming task p is going to be executed in the data process execution instance and the input data of p is the data set$\{d_1,\ldots,d_n\}$, we use PGrap(d) to describe the data provenance of data d, and the data provenance of the process execution instance before p executes can be described as InputP=PGrap(d_1) $\cup\ldots\cup$PGrap(d_n).

Definition (The context of the task at runtime): the context for task p is described as following:

Context(p)={ Contraints(p), **DFDepView**(InputP), **PFDepView**(InputP), **HFDepView**(InputP) }

IV. THE QUERY OPERATIONS FOR DATA PROVENANCE BASED ON DEPENDENCY VIEW

A. Basic query operation

TABLE II. DESCRIPTION OF BASIC QUERY OPERATION

Operation	Query form	Description
Q_1	$\{p\}\ldots Pd$	What tasks directly or indirectly involve in the production of data *Pd*?
Q_2	$\{Pd\}\ldots p$	What data directly or indirectly affect the execution of task *p*?
Q_3	$\{h\}\ldots p$	Who joins the collaboration triggering task *p*?
Q_4	$\{p'\}\ldots p$	What tasks directly or indirectly trigger task *p*?
Q_5	$\{ Pd'\}\ldots Pd$	What data directly or indirectly involve in the production of data *Pd*?
Q_6	$\{h'\}\ldots h$	Who directly or indirectly joins the collaboration with operator *h*?
Q_7	$\{ p \}\ldots h$	What tasks directly or indirectly affect operator *h*?
Q_8	$\{h\}\ldots Pd$	Whose collaborations directly or indirectly affect the production of data *Pd*?
Q_9	$\{ Pd \}\ldots h$	What data are directly or indirectly used by operator *h*?

Based on the dependency views of the task at runtime, the formal expressions of these basic query operations in table II can be expressed as table III.

TABLE III. FORMAL EXPRESSIONS OF BASIC QUERY OPERATION

Operation	Formal expression
$Q_1(Pd)$	$\cup$ {Context(p). **PFDepView** $\cup$ **{ p}**} , where $p\in$ { p' | (Generatedby: $Pd\rightarrow p'$) }
$Q_2(p)$	Context(p). **DFDepView**
$Q_3(p)$	Context(p). **HFDepView**
$Q_4(p)$	Context(p). **PFDepView**

$Q_5(Pd)$	∪{Context(p). **DFDepView**}, where $p \in \{ p' \mid$ (Used: $p' \rightarrow Pd$)}
$Q_6(h)$	∪{Context(p).**HFDepView**}–h, where $p \in \{ p' \mid$ (Controlledby: $p' \rightarrow h$)}
$Q_7(h)$	∪{Context(p).**PFDepView**}, where $p \in \{ p' \mid$ (Controlledby: $p' \rightarrow h$)};
$Q_8(Pd)$	∪{Context(p). **HFDepView** ∪ **{h}** }, where $p \in \{ p' \mid$ (Generatedby: $Pd \rightarrow p'$)} ∧ (Controlledby: $p \rightarrow h$);
$Q_9(h)$	∪{Context(p). **DFDepView** }， where $p \in \{ p' \mid$ (Controlledby: $p' \rightarrow h$)}

B. *Influence query operation*

TABLE IV. DESCRIPTION OF INFLUENCE QUERY OPERATION

Operation	Query form	Description
DataAffectQ$_1$	Pd…{Pd'}	What data directly or indirectly are affected the production of data Pd'?
DataAffectQ$_2$	Pd…{p}	What tasks directly or indirectly are affected by data Pd?
DataAffectQ$_3$	Pd…{h}	Whose collaborations are directly or indirectly are affected by data Pd?
ProcessAffectQ$_1$	p…{p'}	What tasks directly or indirectly are affected by task p' ?
ProcessAffectQ$_2$	p…{ Pd }	What data directly or indirectly are affected by task p ?
ProcessAffectQ$_3$	p…{h}	Whose collaborations are affected by task p?
HumanAffectQ$_1$	h…{h'}	Whose collaborations are affected by h?
HumanAffectQ$_2$	h…{ Pd }	What data are directly or indirectly affected by the collaborations with h?
HumanAffectQ$_3$	h…{p}	What tasks are directly or indirectly affected by the collaborations with h?

Assuming **PDepViewofAll**, **DDepViewofAll** and **CDepViewofAll** stands for the process dependency view, data dependency view and collaboration dependency view of the whole process execution instance, the formal expressions of basic query operations in table IV can be expressed as table V.

TABLE V. FORMAL EXPRESSIONS OF INFLUENCE QUERY OPERATION

Operation	Formal expression
DataAffectQ$_1$(Pd)	**DDepViewofAll** - Q$_5$(Pd)
DataAffectQ$_2$(Pd)	**PDepViewofAll** - Q$_1$(Pd)
DataAffectQ$_3$(Pd)	**CDepViewofAll**- Q$_8$(Pd)
ProcessAffectQ$_1$(p)	**PDepViewofAll** - Q$_4$(p)
ProcessAffectQ$_2$(p)	**DDepViewofAll** - Q$_2$(p)
ProcessAffectQ$_3$(p)	**CDepViewofAll** - Q$_3$(p)
HumanAffectQ$_1$(h)	**CDepViewofAll** - Q$_6$(h)
HumanAffectQ$_2$(h)	**DDepViewofAll** - Q$_9$(h)
HumanAffectQ$_3$(h)	**PDepViewofAll** - Q$_7$(h)

V. QUERY LANGUAGE

Though query operations based on dependency view has improve the efficient of data provenance tracking, we still need to deal with the complexity of the transformation from the requirements of data provenance analysis to the query operations. Therefore we propose a contextual query language base on these query operations.

A. *Data set based on dependency view*

Dependency view is the basic element for the context of task at runtime, the result of basic query operation and the result of influence query operation. Therefore the data related to data provenance tracking is based on dependency view which can be seen as a kind of virtual data table in this paper, and any operation based on dependency view is actually the query to these virtual data tables. Therefore basic query operations, influence query operations and specific dependency views can be divided into three categories from the point view of dependency view, which can carry out the operations to one another internally. The partitions are show as table VI, in which "Instance" stands for the process execution instance.

TABLE VI. DATA SET BASED ON DEPENDENCY VIEW

Dependency view	Corresponding data set
Data dependency view	Instance.Context().DDepViewofAll; Instance.Context(p). DFDepView; Q$_2$, Q$_5$ and Q$_9$; DataAffectQ$_1$, ProcessAffectQ$_2$ and HumanAffectQ$_2$
Process dependency view	Instance.Context().PDepViewofAll; Instance.Context(p). PFDepView; Q$_1$, Q$_4$ and Q$_7$; ProcessAffectQ$_1$, DataAffectQ$_2$ and HumanAffectQ$_3$
Collaboration dependency view	Instance.Context().CDepViewofAll; Instance.Context(p). HFDepView; Q$_3$, Q$_6$ and Q$_8$; HumanAffectQ$_1$, DataAffectQ$_3$ and ProcessAffectQ$_3$

B. *Operators for data set based on dependency view*

Product operation merges one dependency view to another dependency view by eliminating duplicate nodes and edges. Assuming the product of data set t_1 and t_2 is signed as $t_1 \times t_2$, product operation between t_1 and t_2 is described as follow:

$$t_1 \times t_2 = <\mathbf{N} \cup \mathbf{N}', \mathbf{E} \cup \mathbf{E}'>$$

where the corresponding direct graph of t_1 is <**N**, **E**> and the corresponding direct graph of t_2 is < **N**', **E**'>.

Selection operation selects out the tuples satisfied predicate, which use σ to stand for selection operation and set predicate to the subscript of σ. For the data set based on dependency view, there are two differences from relation database. First, the variables in predicate must be the variables defined in dependency view. These variables include: node (standing for data, task and human), edge (standing for the dependency like "Generatedby", "Used", "Triggeredby" and so on), and subgraph (standing for the part of dependency view satisfying specific constraint). Second, it only permit to use the operators including =, ∈ and ⊆. The operators likes $\neq, \leq, \geq, <$ and $>$ are refused. But it can compose the bigger predicate by the operators of single predicate, like ∧, ∨ and ¬. For example, to find the process nodes affected by both p_7 and Pd_2 in figure1 can be described as follow:

$$\sigma_{\text{subGraph} \subseteq WF_2.P_7.\text{Context.PFDepView} \wedge \text{subGraph} \subseteq \text{DataAffectQ}_2(WF_2.Pd_2)}$$
$$(WF_2.\text{Context.PFDepViewAll})$$

Projection operation for data set based on dependency view can be signed by ∏. The subscript of ∏ consists of node, edge and sub graph, meanwhile the rear parameters consists of the data set and the corresponding operators. For example, to find out the tasks which depend on task p_7 can be described as follow:

$$\Pi_{node} WF_2.p_7.\text{Context.PFDepView}$$

C. Syntax of contextual query language

As SQL of relation database, the contextual query language also includes three clauses: select, from and where. Corresponding to projection operator, select clause is used to get the specific elements from process execution instance to satisfy the requirement of data provenance analysis. These elements which are part of context of the task at runtime include: the node (standing for data, task and human), the edge (standing for the dependency) and the sub graph. Corresponding to product operator, from clause is used to list the data set based on dependency view related to the requirement of data provenance analysis. Corresponding to the predicate of selection operator, where clause is used to describe the condition and the constraint for filtering the data set based on dependency view satisfied the requirement of data provenance analysis. Referring to the structure of relation database query language, the many-to-many query of data provenance based on context can be abstracted as follow:

Select $c_1, c_2, \ldots, c_n$ **From** **DepView**$_1$, **DepView**$_2$, …, **DepView**$_m$
Where predicates

DepView$_j$ stands for the data set based on dependency view, and c_i stands for the node, edge, or sub graph in the data set. Therefore the query statement of contextual query language is equivalent to the corresponding data set based on dependency view with their operators, and many-to-many query can also be described as follow:

$$\prod_{c_1,c_2,\cdots c_n}(\sigma_{\text{predicates}}(\mathbf{DepView}_1 \times \mathbf{DepView}_2 \times \cdots \mathbf{DepView}_m))$$

VI. Experiment

To validate the advantage of our query mechanism based on dependency view to the query based on OPM at the aspect of the data provenance tracking, we assess the performance of data provenance query as follow: Firstly, select one node from WF_2 randomly, and then execute query related to the node, which is seen as one test case. Secondly, collect the time cost through the execution of test case as the performance of the query statement. We use the number of database access to stands for the time cost, because the time cost of provenance tracking mainly spend on database access and each database access cost the similar time. Thirdly, set the parameter n as the least number of test case executions, carry out the executions of test case at n times, and increase by 10n times. Finally, sort out the data of performance index of test execution by the parameter n, and then sum the data with the same parameter n.

In this test, we select the complex query requirement "find out the tasks that are affected by the data Pd_2 and also trigger the task p_7" as the test object. Meanwhile, our query language can describe the query requirement as follow query statement:

Select node **From** p_7.Context.PFDepView ∩ DataAffectQ2(Pd_2)

Where the cost of this statement consists of the cost of p_7.Context.PFDepView and the cost of DataAffectQ2(Pd_2).

Set 5 to the parameter n, and the result is shown in figure 2, where the ordinate is the number of database access and the abscissa is the number of the increment of n. In figure 2, the red line stands for the result of query statement, and the blue line stands for the result of regular query based on OPM. We can see that the result of query statement is bigger than the result of the regular query at the beginning, but soon the result of query statement is smaller than the result of the regular query and the gap continues to become wider. The reason is that the dependency view needs more database access than regular query based on OPM to extract the dependency view from the whole data provenance directed graph at the beginning, and then it can support any basic query operation by just one database access. Therefore the experiment shows that the query language based on dependency view is more efficient.

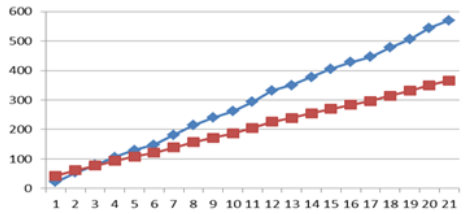

Figure 2. The performance comparing between query statement and corresponding regular query based on OPM

VII. Conclusion

In this paper, we study the query language for the data provenance which keeps growing as the development of data intensive systems and process-aware system. Through the study, we review the challenge and requirement to current data provenance query, and try to find a new language to solve the process analysis problem base on dependency view of data provenance. In the future, we will continue our work to support more and more complex requirements of process analysis based on data provenance.

Acknowledgements

Xin Gao is the corresponding author of this paper.

References

[1] R. Bose and J. Frew. Lineage retrieval for scientific data processing: a survey, ACM Comput. Surv., 37(1): 1-28, 2005.

[2] Y. Simmhan, B. Plale, and D. Gannon. A survey of data provenance in e-science, SIGMOD Record, 34(3): 31–36, 2005.

[3] Muniswamy-Reddy, K., Seltzer, M., Provenance as First-Class Cloud Data, 3rd ACM SIGOPS International Workshop on Large Scale Distributed Systems and Middleware (LADIS'09), October 2009.

[4] Ikeda, R., Park, H., Widom, J.: Provenance for generalized map and reduce workflows. In: CIDR. pp. 273-283 (2011)

[5] Malik, T., Nistor, L., Gehani, A.: Tracking and sketching distributed data provenance. In: eScience. pp. 190-197, 2010.

[6] Chunhyeok Lima, Shiyong Lua, Artem Chebotkob, Farshad Fotouhia, Andrey Kashleva, OPQL: Querying scientific workflow provenance at the graph level, Data & Knowledge Engineering, Volume 88, Pages 37–59, November 2013

[7] Amit Chavan, Silu Huang, Amol Deshpande, Aaron Elmore, Samuel Madden, Aditya Parameswaran,Towards a Unified Query Language for Provenance and Versioning, International Workshop on Theory and Practice of Provenance, 2015.

[8] Grigoris Karvounarakis , Zachary G. Ives , Val Tannen, Querying Data Provenance, SIGMOD, pages: 951-962, 2010

[9] M. K. Anand, S. Bowers, and B. Lud äscher. Techniques for efficiently querying scientific workflow provenance graphs. In EDBT, volume 10, pages 287–298, 2010.

[10] Moreau L, Freire J, Futrelle J, et al. The open provenance model, Southampton: School of Electronics and Computer Science, University of Southampton, 2007.