Provenance-Driven Data Curation Workflow Analysis

Tianhong Song
Department of Computer Science, University of California, Davis
One Shields Avenue, Davis, CA, USA (530) 752-7004
thsong@ucdavis.edu
Advisor: Bertram Ludäscher

ABSTRACT
Manually designed workflows can be error-prone and inefficient. Workflow provenance contains fine-grained data processing information that can be used to detect workflow design problems. In this paper, we propose a provenance-driven workflow analysis framework that exploits both prospective and retrospective provenance. We show how provenance information can help the user gain a deeper understanding of a workflow and provide the user with insights into how to improve workflow design.

Categories and Subject Descriptors
H.4.1 [**Information Systems Applications**]: Office Automation – *Workflow Management;*
H.3 [**Information Storage and Retrieval**]: Miscellaneous

General Terms
Management, Design.

Keywords
Workflow Analysis; Workflow Design; Data Curation

1. INTRODUCTION
Electronic data is rife with quality issues like inaccuracy, inconsistency or incompleteness [2]. Therefore, data curation is increasingly important in data sharing, integration and use. Workflow technologies can be employed to automate and facilitate data-curation pipelines using workflow management systems [1]. On the other hand, given a library of data-curation actors, the workflow design process can be time-consuming and error-prone, especially for scientists with little programming experience. Further, correcting problematic workflows or improving inefficient workflows once again relies on the user's programming abilities. Usually, the user, like a data collection manager, has a dataset in hand but does not have a clear idea on how to design and develop a workflow to perform certain data-curation tasks. Thus, the user tries to put together a working workflow and runs it against some test datasets multiple times in order to modify and improve the workflow until it can accomplish expected data-curation tasks.

However, this approach can be inefficient and limited because 1) the result of a workflow run and the workflow graph itself may not contain enough information that can provide the user with insights into how to correct or improve the workflow; 2) a workflow may contain some design problems or can be designed in a more efficient way but the user may not be aware of those issues; and 3) data-driven workflows such as data-curation workflows are highly data intensive and so running a data-curation workflow on large datasets or even test datasets is time-consuming and inefficient. Collection-oriented workflow models provide nested data model and scope configuration which have proven advantageous for workflow design, e.g., improving reusability, simplicity, predictability and ease of use [22]. Nonetheless, the workflow design process is not necessarily simplified since some of the fine-grained data dependency information is hidden from workflow graphs and the user needs to link workflow graphs with actor configurations and input data in order to fully understand the data flow of a workflow and find and correct possible workflow design problems.

In order to tackle these and related problems, we propose a provenance-driven data-curation workflow analysis framework that helps the user build a problem-free and optimized workflow more efficiently. Specifically, the proposed framework provides the following features:

- Perform analysis on the abstract provenance of a workflow to detect workflow design problems ahead of execution
- Detect problems and possible errors in the result of a workflow run by exploiting runtime provenance
- Link runtime provenance to abstract provenance to detect unexpected runtime behaviors of a workflow

We show that our proposed framework improves on the current approaches from the following aspects: 1) analysis and querying on the runtime provenance can help the user better understand the result and find possible problems of a workflow run, therefore, this provides insights into correcting or improving the workflow; 2) workflow analysis can help the user detect workflow design issues and suggest ways to improve; 3) since the user can get a better understanding of a workflow through workflow analysis, less tests need to be run and some of tests can even be avoided since static analysis can be performed to detect possible design problems ahead of execution. Our previous work [23] has proposed some ideas to perform better workflow analysis and optimization on data-curation workflows using abstract provenance graphs [30]. This paper formalizes and extends our previous approach with analysis on both prospective and retrospective provenance and the interaction between them. As a research proposal, we introduce the terminologies we use, describe how the proposed framework works, and give examples to demonstrate the ideas in this paper.

2. DATA CURATION EXAMPLE
Data curation involves an active and on-going management of data throughout its lifecycle. One of the key components of data curation is improving and maintaining data quality and workflows can be used to automate such processes. Here, we show an

SIGMOD'15 PhD Symposium, May 31, 2015, Melbourne, Victoria, Australia.

ACM 978-1-4503-3529-4/15/05…$15.00.
http://dx.doi.org/10.1145/2744680.2744691

(a)

Record Id	Event Date	Recorded	State Pro	Scientific Name	Scientific Name Aut	Decimal Lat	Decimal Lon	Year	Family	Reproductive C	Locality
SCAN.11364	1972-08-13	Roman S.	Arizona	Euptoieta claudia	(Cramer, 1775)	34.052139	-109.729612	1972	Nymphalidae		Intersecti
BPBM.478988	1965-01-00	Hoe, W.J.	Hawaii	Fissidens bryoides	Hedwig, 1801	21.58667	-157.91306	1965	Fissidentace	Flower not preser	Koolau M
BPBM.481599	1965-01-03	Hoe, W.J.	Hawaii	Rhizogonium S.E	Bridel, 1827	21.30694	-157.765	1965	Rhizogoniace	Fruiting	1 mi. S o
SCAN.436021	1966-04-10	J.H. Davids	Arizona	Gymnopyge hoplia	Linell, 1896	-114.217126	33.615386	1966	Scarabaeidae		4 mi. S o
SCAN.13224		Roman S.	Arizona	Speranza trilineari	(Grossbeck, 1910)	33.96222	-111.84833	1968	Geometridae		Seven Sp

(b)

Record Id	Event Date	Recorded	State Pro	Scientific Name	Scientific Name Aut	Decimal Lat	Decimal Lon	Year	Family	Reproductive C	Locality
SCAN.11364	1972-08-13	Roman S.	Arizona	Euptoieta claudia	**Cramer, 1775**	34.052139	-109.729612	1972	Nymphalidae		Intersec
BPBM.478988	1965-01-00	Hoe, W.J.	Hawaii	Fissidens bryoides	Hedwig, 1801	21.58667	-157.91306	1965	Fissidentace	Flower not preser	Tempe
BPBM.481599	1965-01-03	Hoe, W.J.	Hawaii	Rhizogonium S.E	Bridel, 1827	21.30694	-157.765	1965	Rhizogoniace	Fruiting	1 mi. S
SCAN.436021	1966-04-10	J.H. Davids	Arizona	Gymnopyge hoplia	**Linell, 1895**	-114.217126	33.615386	1966	Scarabaeidae		4 mi. S
SCAN.13224	**1968-05-18**	Roman S.	Arizona	Speranza trilineari	(Grossbeck, 1910)	33.96222	-111.84833	1968	Geometridae		Seven S

Figure 1. Example collection dataset: (a) the original dataset with data quality issues, (b) the expected result after data curation

example of data curation that deals with the problem of data incompleteness. Imagine a user such as a natural science collection data manager who curates datasets that are typically standardized using the Darwin Core [28] term set as a data curator. The user wants to run a data-curation workflow on a dataset to check the validity of the dataset and curate data quality issues if any exist. An example dataset is shown in Figure 1(a), and the expected result after running some data-curation tasks is shown in Figure 1(b). Note that some of the values in the result are changed. For example, the "Event Date" of the record with id "SCAN.13224" is changed from an empty value to "1968-05-18". The updated value is constructed from atomic data attributes including "Year", "Month" and "Day" ("Month" and "Day" not shown). The "Event Date" of this record is no longer empty after data curation, which is an improvement on the completeness of this record.

3. PRELIMINARIES, TERMINOLOGIES

3.1 Workflow Model

Typically, the input to a data-curation workflow is structured as a data stream, which consists of a set of records. Each record contains a set of attribute-value pairs (or data items). In our workflow model, a workflow consists of interconnected actors; connections between actors indicate how data flows. Each data validation actor can have a "black-box" and a configuration. The black-box implements the actual data-processing logic, which is wrapped with a well-defined configuration that includes a set of "read scope" functions for selecting relevant parts of the input data stream and a set of "write scope" functions for combining the result of the black-box with the unselected part of the data stream to form the output of the actor. Each black-box has a set of input ports and a set of output ports. For each data item read by a black-box on its input ports, the black-box either validates the data item and updates it or reads this data item only for reference during validation of other data items. The unselected part of the data stream is transported, bypassing the black-box. Note that workflows in our simplified model are linear by default, i.e., they exclude branching and feedback loops. Such features can be handled implicitly within actors if needed, or added to the framework as in COMAD [20]. More details about the workflow model can be found in [23].

3.2 Provenance Model

Provenance contains information about the origin context, derivation, ownership, or history of data artifacts [7]. There are two forms of provenance: prospective and retrospective provenance [8]. Prospective provenance, or abstract provenance, which captures the specification of computational tasks, indicates how data products should be generated. Abstract provenance can be constructed from workflow specification, actor configuration and input data schema. Retrospective provenance, or runtime provenance, which captures the workflow execution information, indicates how data products are derived during runtime. In our model, runtime provenance is captured at attribute level and is attached to each record. That is, the provenance information about a data-curation task (i.e., an invocation of a data-curation actor on a record) is stored as a separate set of data attributes that will be added to the record. We propose a graph-based provenance model including two types of provenance graphs: Abstract Data Dependency Graphs (ADG) and Concrete Data Dependency Graphs (CDG), which capture prospective and retrospective provenance respectively.

An ADG is a type of abstract provenance graph [30]. An ADG has a set of nodes, N, that represent data items and a set of edges, E, that represent dependencies among data items. Since different types of dependencies are created by actors, each edge has a label, D:A, that indicates what type of dependency, D, the edge represents and which actor, A, creates this dependency. An example ADG is shown in Figure 2(b) (actor name not shown), and its corresponding workflow is shown in Figure 2(a). A CDG captures actual data dependencies created during a workflow run. In a CDG, we use a graph notation similar to that in an ADG, except that each node in a CDG has a label, T:V:S, that indicates the data type, T, the data value, V, and the curation status, S, of the data item. Note that one CDG is associated with one record since one record in the result of a workflow run can be seen as one invocation of the workflow. Figure 2(c) shows an example CDG (actor name and data type not shown) of the record with id "SCAN.436021" in Figure 1(b). In some of the following examples, for simplicity, the data type of nodes in a CDG and the actor name of edges in a ADG or CDG are not shown (e.g., "1966-04-10 : Unable_Curate" instead of "EventDate : 1966-04-10 : Unable_Curate" and "update" instead of "Event Date Validator : Update"). Different background colors of data items in a CDG indicate the curation status of a data item. We have identified four different curation statuses:

• Valid (Green): The data item is valid after checking against a standard.

• Curated (Blue): Issues have been found in the data item or the original value was missing, and this data item has been updated with a corrected (valid) value.

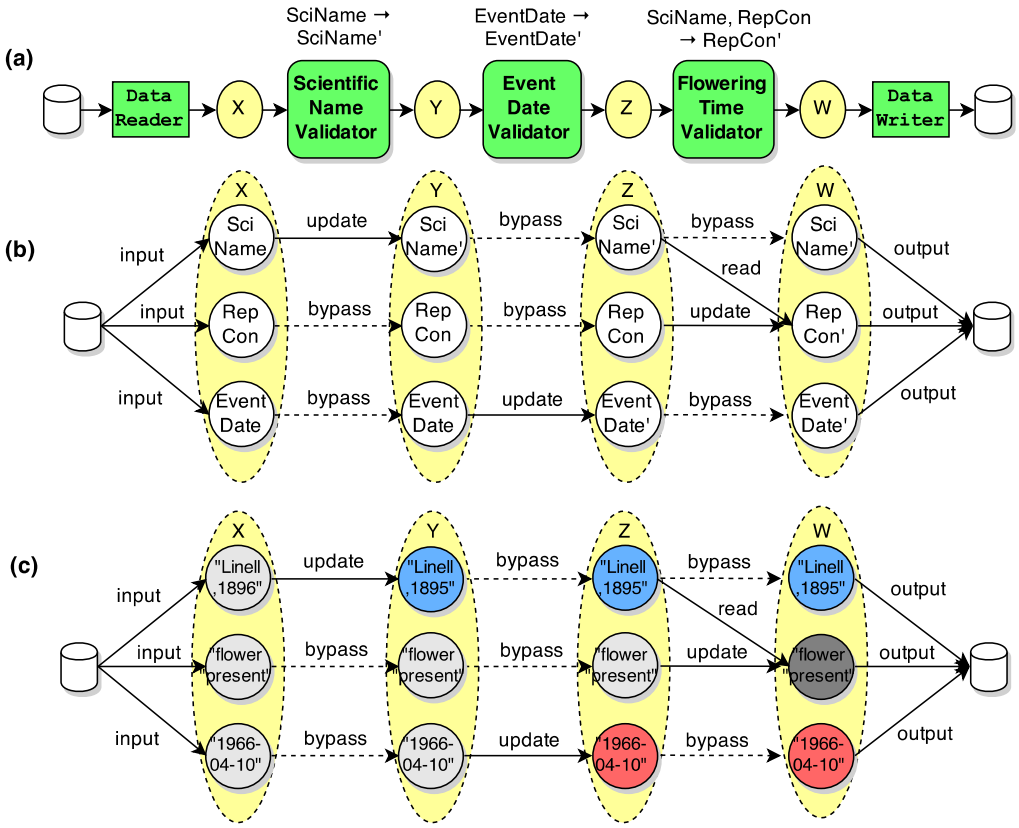

Figure 2. (a) Example data-curation workflow with three actors; (b) Example ADG of the workflow (c) Example CDG of a workflow run

• Unable_Curate (Red): Issues have been found in the data item or the original value was missing, but the system failed to find a valid value to update it with.

• Unable_Detemine_Validity (Grey): The workflow failed to make an assertion on the validity of the data item due to lack of information or system errors.

3.3 Workflow Patterns and Graph Query

In order to perform workflow analysis, we use data-curation workflow patterns to encode workflow design problems. We use the term *workflow anti-patterns* to represent design issues. That is, if a workflow contains a certain anti-pattern, then the workflow has the design issue this anti-pattern represents. In contrast, a *workflow pro-pattern* is a type of workflow pattern that a workflow must contain. For example, a workflow should be connected in most cases, thus a pro-pattern "The workflow is connected" can be enforced in such cases to make sure the workflow is connected.

In the proposed framework, analysis is performed based on a set of selected anti-patterns and pro-patterns. Once invoked, a set of built-in workflow anti-patterns and pro-patterns is provided to the user. Those patterns can be selected and then enforced in the workflow analysis. If a provenance graph contains a selected anti-pattern or doesn't contain a selected pro-pattern, the matching anti-pattern will be highlighted in the visualized graph or the missing pro-pattern will be returned. Any other workflow patterns that are not selected are allowed implicitly by default.

As our first prototype, we define workflow patterns using Regular Path Query (RPQ) [29] which can be used to query provenance graphs. We adopt Regular Data Path Query (RDPQ) [18] as an extension of RPQ, which can be used to query not only edges but also nodes in a provenance graph. We also adopt subgraph matching algorithm [11] to find out whether two provenance graphs conflict or not.

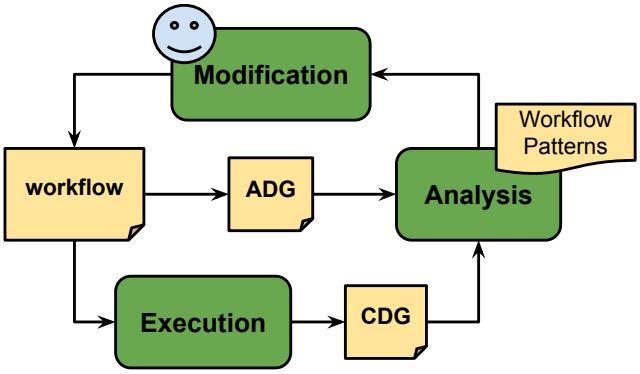

Figure 3. Overview of the proposed data-curation workflow analysis framework

4. PROPOSED WORKFLOW ANALYSIS FRAMEWORK

An overview of the proposed workflow analysis framework is shown in Figure 3. Starting with a workflow as the input to the framework, workflow analysis is performed on the provenance information of the input workflow, which helps the user find possible workflow design problems in order to improve workflow

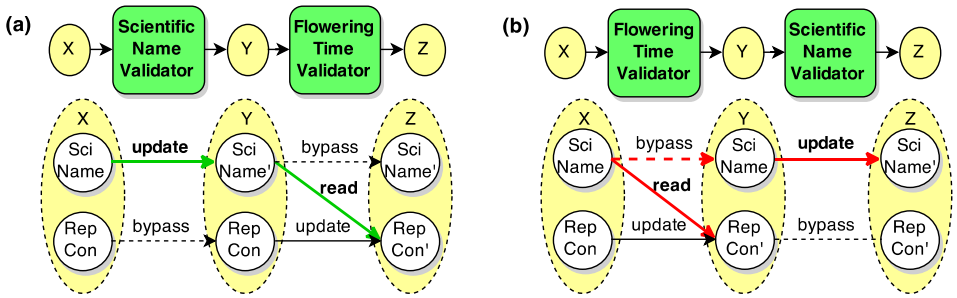

Figure 4. Example ADG analysis where certain order of actors must be maintained

design. The analysis component shown in the figure can be invoked with an ADG, a CDG or both. A set of built-in workflow patterns is provided by the framework for the user to choose from. The user can also define their own workflow patterns. Our framework also provides a GUI that visualizes workflows and provenance graphs showing possible design problems, which allows the user to modify and interact with workflows.

4.1 ADG Analysis

In our proposed framework, static analysis is performed on a given workflow in order to detect design problems ahead of workflow execution. Some of the fine-grained data dependency information is hidden from a workflow graph, which can be discovered by constructing an ADG. An ADG can be constructed by combining workflow graph, actor configurations and input dataset schema. A set of selected workflow patterns will be used to query the constructed ADG to detect workflow design issues.

Example. In some cases, a particular order of the actors in a workflow needs to be maintained. Figure 4 shows two different orders of the actors in a workflow. In this example, "Flowering Time Validator" reads "SciName" as a reference to validate "RepCon", so "SciName" should be validated first in order to avoid generating incorrect results. An anti-pattern "Update After Read" can be enforced to detect this issue, which can be formulated as "read^{-1}.bypass*.update". The ADG shown in Figure 4(b) has a match for that query, which indicates that the workflow has the actor order issue this query represents.

4.2 CDG Analysis

By looking at the result of a workflow run alone (e.g., the example result in Figure 1(b)), the user may find it difficult to understand how the result is derived and why the workflow asserts certain curation in the result. In the proposed framework, not only the result of a workflow run but also the runtime provenance are given back to the user after a workflow execution, i.e., a CDG is constructed, which gives the user more information about the result. A set of workflow patterns can be selected and enforced such that workflow runtime problems can be detected by querying the constructed CDG.

Example. In some cases, even though a data item has a curation status "Valid", it can be a "false positive" case. This may be because if an actor reads a data item that has an "Unable_Curate" status during invocation, then the output result could be problematic. To detect this problem, an anti-pattern "Green After Red" can be enforced to check whether among all the data items in the result that have a "Valid" status, there exist some data item that used a reference data item with a "Unable_Curate" status during curation. This pattern can be formulated as

$$\exists(i,j) \in R_{RDPQ} = \text{read}.(any.\text{bypass})^*.\text{Valid}.\text{output},$$
$$\rho(i) = \text{Unable\_Curate}^1$$

Figure 5(b) shows an example CDG that has a match of this anti-pattern, which indicates that the result may have issues and the user may need to modify the workflow in order to avoid this problematic result.

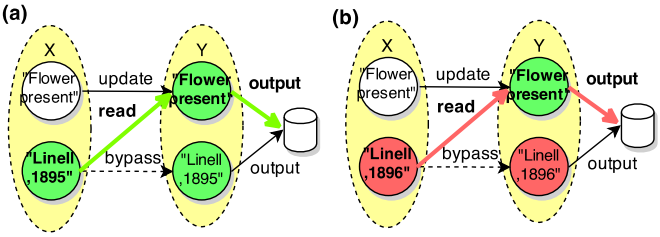

Figure 5. Example CDG analysis where incorrect result is yielded due to invalid reference

4.3 ADG and CDG Interaction

An ADG can be seen as a prediction of the runtime behavior of a workflow, and a CDG contains information about the actual runtime behavior of a workflow. If the runtime behavior of a workflow is as expected, then the CDG of a workflow should have the same structure as the ADG of the workflow. Otherwise, some unexpected problems happened during runtime. Therefore, some workflow runtime problems can be discovered by linking CDGs with ADGs, i.e., comparing prospective and retrospective provenance. There are different reasons why the ADG and the CDG of a workflow have different structures. For example, some actor is misconfigured (i.e., different than the actor declaration), or some of the actors are not invoked during runtime due to missing input data items or incompatible data types. In the proposed framework, we use subgraph-matching algorithms to find out whether the CDG matches the ADG of a workflow. If conflict exists, the differences between a CDG and an ADG are highlighted, which gives the user insights into why unexpected problems occurred during runtime.

Example. Figure 6 shows a conflict between the CDG and the ADG of a workflow due to incorrect configuration of "Flowering Time Validator". As the declaration of this actor indicates, the

[1] R_{RDPQ} is the result of evaluating a regular data path query. "read", "bypass" and "output" are edge labels (type of data dependency) and "any" and "Valid" are node labels (curation status of data items) in a CDG. $\rho: N \rightarrow S$, is a function that maps a node to its curation status.

actor should read “SciName” and “RepCon”. But the actor may be misconfigured in the way that it read the data item labeled “EventDate” instead of “SciName” during runtime, which yielded incorrect dependencies during runtime. Therefore, the result of this workflow run is likely to have problems.

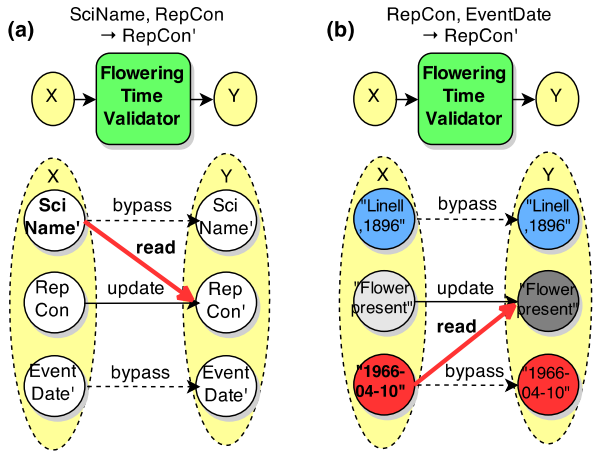

Figure 6. Example ADG and CDG conflict due to configuration error

4.4 Implementation

We have developed our first prototype of the analysis component in the proposed framework in Datalog using the DLV system [17], with a wrapper written in Java as the interface to interact with other components in the framework. Workflow specification, provenance graphs and workflow patterns are encoded as Datalog rules. A RPQ evaluation tool in Datalog [24] is used to evaluate workflow patterns on provenance graphs.

5. RELATED WORK

Some existing workflow systems like Kepler [19] and VisTrails [6] for example provide user friendly GUIs where workflows are visualized as data-flow graphs and the user can easily understand the structure of the workflow and make changes accordingly. Some workflow design steps are automated in some workflow systems like Wings [15], and several other frameworks [4, 10] has been proposed to facilitate workflow design. Graph-based workflow analysis has been widely studied [9], as well as in the domain of business process management [5] and process-aware information systems [27]. Various aspects of runtime workflow analysis including real time analysis [16] and execution monitoring [25] for example have been studied. Efficient provenance storage and querying techniques have been proposed for workflows [13] that makes it possible for the user to interpret, validate, debug and reproduce workflow execution by querying provenance [3]. However, little work has been done for data-curation workflows or collection-oriented workflows [21] in general. And to the best of our knowledge, our proposed framework is the first work that combines retrospective and prospective provenance to perform workflow analysis. In this way, some of the workflow design problems that may be hidden by exploiting one type of provenance can be discovered. Existing data-curation workflow packages or systems like Kurator 1.0 [14], BioVel [26] and Nadeef [12] for example provide a library of data-curation actors and workflows for the user to choose from. However, little support for workflow design and analysis is provided. Even modifying an existing curation workflow requires detailed knowledge about the underlying workflow system.

6. CONCLUSION AND FUTURE WORK

We propose a provenance-driven data-curation workflow analysis framework that helps the user develop and improve data-curation workflows more efficiently. Both prospective and retrospective provenance is exploited in order to capture fine-grained data dependency information of a workflow. Graph queries and subgraph matching algorithms are applied to analyze provenance graphs in order to detect workflow design problem.

Our first prototype only contains limited functionalities (e.g., small number of encoded patterns). We will continue developing our framework and one of our future works is to develop (semi-) automated workflow pattern discovery mechanism through the analysis process. At the same time, we simplify our workflow model to a common use case but the model is restricted in the way that some of the use cases cannot be handled. So our future work will include generalizing the workflow model to make the framework compatible with more use cases.

7. REFERENCES

[1] Barker, A. and Hemert, J. Van Scientific workflow: a survey and research directions. *Parallel Processing and Applied Mathematics*. (2008), 746–753.

[2] Batini, C. and Scannapieco, M. *Data quality: concepts, methodologies and techniques*. Springer, 2006.

[3] Bowers, S. Scientific Workflow, Provenance, and Data Modeling Challenges and Approaches. *Journal on Data Semantics*. 1, 1 (Apr. 2012), 19–30.

[4] Bowers, S. and Ludäscher, B. Actor-oriented design of scientific workflows. *Lecture Notes in Computer Science*. 3716, (2005), 369–384.

[5] Brogi, A., Corfini, S. and Popescu, R. Semantics-based composition-oriented discovery of Web services. *ACM Transactions on Internet Technology*. 8, 4 (Sep. 2008), 1–39.

[6] Callahan, S.P., Freire, J., Santos, E., et al. VisTrails: visualization meets data management. *Proceedings of the 2006 ACM SIGMOD international conference on management of data*. (2006), 745–747.

[7] Cheney, J., Finkelstein, A., Ludäscher, B., et al. Principles of Provenance (Dagstuhl Seminar 12091). *Dagstuhl Reports*. 2, 2 (2012).

[8] Clifford, B. and Foster, I. Tracking provenance in a virtual data grid. *Concurrency and Computation: Practice and Experience*. 20, 5 (2008), 565–575.

[9] Cohen-Boulakia, S., Chen, J., Missier, P., et al. Distilling structure in Taverna scientific workflows: a refactoring approach. *BMC bioinformatics*. 15, Suppl 1 (Jan. 2014), S12.

[10] Consortium, B. Interoperability with Moby 1.0—It’s better than sharing your toothbrush! *Briefings in bioinformatics*. 9, 3 (2008), 220–231.

[11] Cordella, L. and Foggia, P. A (sub) graph isomorphism algorithm for matching large graphs. *Pattern Analysis and Machine Intelligence, IEEE Transactions*. 26, 10 (2004), 1367–1372.

[12] Dallachiesa, M., Ebaid, A., Eldawy, A., et al. NADEEF: a commodity data cleaning system. *Proceedings of the 2013 ACM SIGMOD international conference on management of data* (New York, New York, USA, Jun. 2013), 541–552.

[13] Davidson, S.B. and Freire, J. Provenance and scientific workflows: challenges and opportunities. *Proceedings of the*

2008 ACM SIGMOD international conference on management of data (2008), 1345–1350.

[14] Dou, L., Cao, G., Morris, P.J., et al. Kurator: A Kepler package for data curation workflows. *Procedia Computer Science*. 9, (Jan. 2012), 1614–1619.

[15] Gil, Y., Ratnakar, V. and Kim, J. Wings: Intelligent workflow-based design of computational experiments. *IEEE Intelligent Systems*. (2010), 62–72.

[16] Gunter, D., Deelman, E., Samak, T., et al. Online workflow management and performance analysis with stampede. *Proceedings of the 7th International Conference on Network and Services Management*. (2011), 152–161.

[17] Leone, N., Pfeifer, G., Faber, W., et al. *The DLV system*.

[18] Libkin, L. and Vrgoc, D. Regular path queries on graphs with data. *Proceedings of the 15th International Conference on Database Theory*. (2012), 74–85.

[19] Ludäscher, B., Altintas, I., Berkley, C., et al. Scientific workflow management and the Kepler system. *Concurrency and Computation: Practice and Experience*. 18, 10 (Aug. 2006), 1039–1065.

[20] McPhillips, T. and Bowers, S. An approach for pipelining nested collections in scientific workflows. *ACM SIGMOD Record*. 34, 3 (Sep. 2005), 12–17.

[21] McPhillips, T., Bowers, S. and Ludäscher, B. Collection-oriented scientific workflows for integrating and analyzing biological data. *Data Integration in the Life Sciences*. 4075, (2006), 248–263.

[22] McPhillips, T., Bowers, S., Zinn, D., et al. Scientific workflow design for mere mortals. *Future Generation Computer Systems*. 25, 5 (2009), 541–551.

[23] Song, T., Köhler, S. and Ludäscher, B. Towards automated design, analysis and optimization of declarative curation workflows. *International Journal of Digital Curation*. 9, 2 (2014), 111–122.

[24] Tekle, K.T., Gorbovitski, M. and Liu, Y.A. Graph queries through datalog optimizations. *Proceedings of the 12th international ACM SIGPLAN symposium on Principles and practice of declarative programming* (New York, New York, USA, Jul. 2010), 25–34.

[25] Vahi, K., Harvey, I., Samak, T., et al. A general approach to real-time workflow monitoring. *High Performance Computing, Networking, Storage and Analysis (SCC)*. (2012), 108–118.

[26] Vicario, S., Hardisty, A. and Haitas, N. BioVeL: Biodiversity virtual e-Laboratory. *EMBnet.journal*. 17, 2 (Sep. 2011), 5–6.

[27] Weber, B., Reichert, M. and Rinderle-Ma, S. Change patterns and change support features–enhancing flexibility in process-aware information systems. *Data & knowledge engineering*. 66, 3 (2008), 438–466.

[28] Wieczorek, J., Bloom, D., Guralnick, R., et al. Darwin Core: an evolving community-developed biodiversity data standard. *PloS one*. 7, 1 (Jan. 2012), e29715.

[29] Wood, P.T. Query languages for graph databases. *ACM SIGMOD Record*. 41, 1 (Apr. 2012), 50–60.

[30] Zinn, D. and Ludäscher, B. Abstract provenance graphs: anticipating and exploiting schema-level data provenance. *Provenance and Annotation of Data and Processes*. (2010), 206–215.