Capturing Interactive Data Transformation Operations Using Provenance Workflows

Tope Omitola[1](✉), André Freitas[2], Edward Curry[2], Séan O'Riain[2], Nicholas Gibbins[1], and Nigel Shadbolt[1]

[1] Web and Internet Science (WAIS) Research Group Electronics and Computer Science, University of Southampton, Southampton, UK
{tobo,nmg,nrs}@ecs.soton.ac.uk

[2] Digital Enterprise Research Institute (DERI), National University of Ireland, Galway, Ireland
{andre.freitas,ed.curry,sean.oriain}@deri.org

Abstract. The ready availability of data is leading to the increased opportunity of their re-use for new applications and for analyses. Most of these data are not necessarily in the format users want, are usually heterogeneous, and highly dynamic, and this necessitates data transformation efforts to re-purpose them. Interactive data transformation (IDT) tools are becoming easily available to lower these barriers to data transformation efforts. This paper describes a principled way to capture data lineage of interactive data transformation processes. We provide a formal model of IDT, its mapping to a provenance representation, and its implementation and validation on Google Refine. Provision of the data transformation process sequences allows assessment of data quality and ensures portability between IDT and other data transformation platforms. The proposed model showed a high level of coverage against a set of requirements used for evaluating systems that provide provenance management solutions.

Keywords: Linked Data · Public open data · Data publication · Data consumption · Semantic Web · Workflow · Extract-Transform-Load · Provenance · Interactive data transformation

1 Introduction

The growing availability of data on the Web and in organizations brings the opportunity to reuse existing data to feed new applications or analyses. In order to reuse existing data, users must perform data transformations to repurpose data for new requirements. Traditionally, data transformation operations have been supported by data transformation scripts organized inside ETL (Extract-Transform-Load) environments or by ad-hoc applications. Currently, users developing data transformation programs follow a typical software development cycle, taking data samples from datasets, and developing the transformation logic, testing and debugging. These approaches are problematic in emerging scenarios such

E. Simperl et al. (Eds.): ESWC 2012 Satellite Events, LNCS 7540, pp. 29–42, 2015.
DOI: 10.1007/978-3-662-46641-4_3

as the Linked Data Web where the reduction of the barriers for producing and consuming new data brings increasing challenges in coping with heterogeneous, high-volume, and dynamic data sources. In these scenarios, the process of *interactive data transformation* emerges as a solution to scale data transformation.

Recently platforms, such as Google Refine[1], are exploring user interface and interaction paradigms for defining data transformations. These platforms allow users to operate over data using Graphical User Interface (GUI) elements instead of scripts for data transformation. By providing a set of pre-defined operations and instant feedback mechanism for each iteration of the data transformation process, this model defines a powerful approach to data transformation and curation. However, despite its practical success, the assumptions behind platforms such as Google Refine have not been modeled nor formalized in the literature. As the need for curation increases in data abundant environments, the need to model, understand, and improve data curation and transformation processes and tools emerges as a critical problem. A foundational model of IDT that (a) brings out the underlying assumptions that IDT platforms use, (b) makes explicit how IDT relates to provenance, and (c) helps IDT platforms be comparable and thereby helping in defining interfaces for their interoperability, would be highly useful.

The process of data transformation is usually positioned as one element in a connected pipeline which needs to be integrated with different processes, to form a **workflow**. Currently, the set of transformation operations present in IDT platforms are not materialized in a way that could allow its use in contexts outside the IDT environment. A transformation process normally involves a data object that is being transformed, and the transformation procedure itself. *Provenance* is the contextual metadata describing the origin or source of that data. *Prospective provenance* provides mechanisms to describe generic workflows which could be used to materialize (future) data transformations. *Retrospective provenance* captures past workflow execution and data derivation information to provide useful context for what had happened up to the present state and time.

This paper investigates the two complementary perspectives described above, and an approach for expressively capturing and persisting provenance in IDT platforms. A provenance extension for the Google Refine platform is implemented and used to validate the proposed solution. The supporting provenance model focuses on the maximization of interoperability, using the three-layered data transformation model proposed in [6] and uses Semantic Web standards to persist provenance data.

The contributions of this work include: (a) a formal model of IDT; (b)the use of an ontology-based provenance model for mapping data transformation operations on IDT platforms; (c) the verification of the suitability of the proposed provenance model for capturing IDT transformations, using Google Refine as the IDT platform; and (d) the validation of the system against the set of requirements, from the literature, used for evaluating provenance management provided by IDT platforms.

[1] http://code.google.com/p/google-refine/.

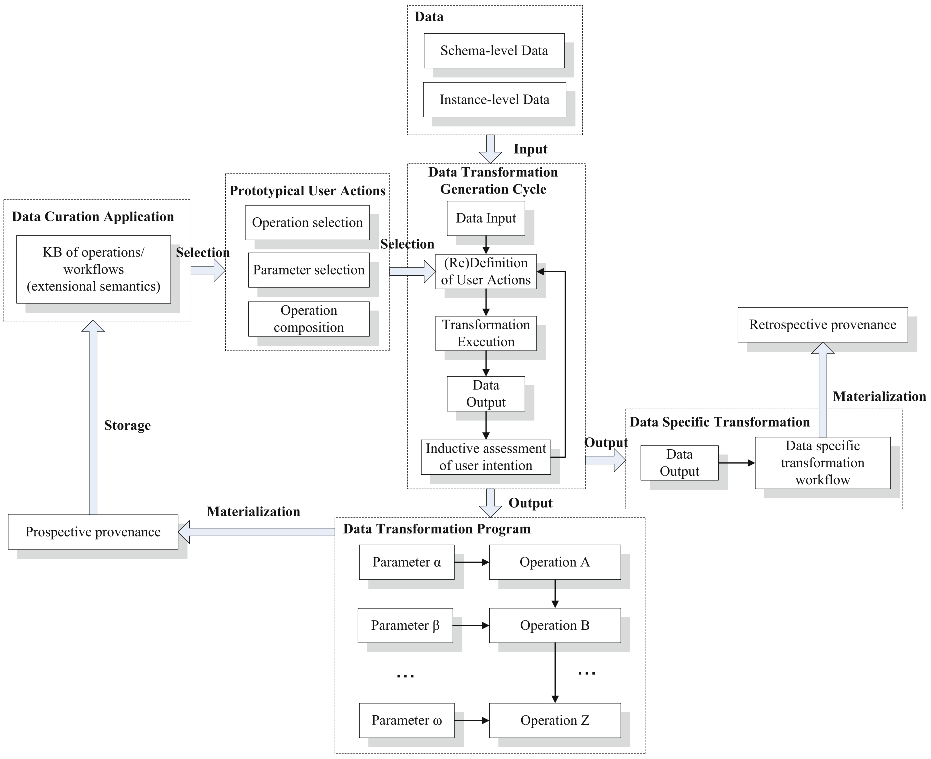


Fig. 1. A model of interactive data transformation

2 Interactive Data Transformations (IDT)

2.1 IDTs, ETL, and Data Curation

Governments across the globe are making their data available in a single Web-based dataspace, such as data.gov.uk, visible across different government agencies, which may also include third-party open data. In this data-rich and heterogeneous environment, data consumers and producers face major data management challenges. Data transformation tasks such as data cleaning, refinement, aggregation, and analysis are usually mediated by programming tasks (in the form of ETL scripts or in generic data transformation programs), making these transformations time consuming and expensive. The increasing availability of third-party datasets brings potential challenges to the centralized nature of traditional ETL environments. Analytical tasks are likely to strongly benefit from data which is present in large numbers of datasets, increasing the cost of the data transformation tasks. With more datasets (such as the open data Web, third-party datasets, etc.) at one's disposal, a large number of ad-hoc small data transformation tasks may emerge.

The concept of interactive data transformation is strongly related to data curation, where human intervention in the data aggregation, cleaning, and transformation is increased. According to Buneman et al. [1], "curated databases are databases that are populated and updated with a great deal of human effort". Curry et al. [2] defines data curation as "the active management and appraisal of data over its life-cycle of interest ... data curation is performed by expert curators responsible for improving the accessibility and quality of data". In a scenario where third-party data becomes abundant on the Web, the process of curating the data for a new intended use becomes an intrinsic part of the data consumption process.

IDTs naturally map to a workflow structure which makes its representation isomorphic to another critical dataspace concern: the capture and representation of provenance. In dataspaces, when datasets cross their original context of use (a specific government agency, for example), data consumers need additional information to determine the trustworthiness and general suitability of the data. For example, it will be very useful for the provenance data representing the processes, artifacts, and agents behind these data transformations to be made available so that data consumers can determine its quality (i.e. answering queries such as *Who modified this value?, From which table is the data derived?*, etc.). The interplay between IDT platforms and provenance addresses important challenges emerging in this new data environment, facilitating the process of data transformation and blurring the borders between ETL and IDT.

2.2 IDT Operations Overview

IDT is defined as the application of a pre-defined set of data transformation operations over a dataset. In IDT, after a transformation operation has been selected from the set of available operations (**operation selection**), users usually need the input of configuration parameters (**parameter selection**). In addition, users can compose different sets of operations to create a data transformation workflow (**operation composition**). Operation selection, parameter selection, and operation composition are the core user actions available for interactive data transformation as depicted in Fig. 1.

In the IDT model, the expected outputs are a data transformation program and a transformed data output (Fig. 1). The transformation program is generated through an iterative *data transformation program generation cycle* where the user selects and composes an initial set of operations, executes the transformation, and assesses the suitability of the output results by an inductive analysis over the materialized output. This is the point where users decide to redefine (*reselect*) the set of transformations and configuration parameters. The organization of operations as GUI elements minimizes program construction time by minimizing the overhead introduced by the need to ensure programming language correctness and compilation/deployment.

The transformation generation cycle generates two types of output: a data output with a data specific transformation workflow, and a data transformation program which is materialized as a prospective provenance descriptor. A **provenance descriptor** is a data structure showing the relationship between the inputs, the outputs, and the transformation operations applied in the process. While the provenance-aware data output is made available for further changes, the prospective provenance workflow is inserted into the KB of available workflows, and can be later reused.

In the next section, we shall present an algebra for Interactive Data Transformation.

2.3 Foundations - an Algebra for Provenance-Based Interactive Data Transformation (IDT)

Despite the practical success of IDT tools, such as Google Refine, for data transformation and curation, the underlying assumptions and the operational behaviour behind such platforms have not been explicitly brought out. Here, we present an algebra of IDT, bringing out the relationships between the inputs, the outputs, and the functions facilitating the data transformations.

Definition 1: Provenance-based Interactive Data Transformation Engine $\mathfrak{G}$. A provenance-based Interactive Data Transformation Engine, $\mathfrak{G}$, consists of a set of transformations (or activities) on a set of datasets generating outputs in the form of other datasets or events which may trigger further transformations.

$\mathfrak{G}$ is defined as a tuple,

$$\mathfrak{G} \ = < \mathbb{D}, (D \cup V), \mathbb{I}, \mathbb{O}, \Sigma, \sigma, \lambda >$$

where

1. $\mathbb{D}$ is the non-empty set of all datasets in $\mathfrak{G}$,
2. D is the dataset being currently transformed,
3. V is the set of views in $\mathfrak{G}$ (V may be empty),
4. $\mathbb{I}$ is a finite set of input channels (this represents the points at which user interactions start),
5. $\mathbb{O}$ is a finite set of output channels (this represents the points at which user interactions may end),
6. Σ is a finite alphabet of actions (this represents the set of transformations provided by the data transformation engine),
7. σ is a finite set of functions that allocate alphabets to channels (this represents all user interactions), and
8. $\lambda = < \ \mathbb{D} \times \mathbb{O} \rightarrow \Sigma \ >$ is a function, where, in a modal transformation engine, $\lambda(D, O) \in \sigma(O)$ is the dataset that is the output on channel O when D is the dataset being currently transformed.

Definition 2: Interactive Data Transformation Event. An Interactive Data Transformation Event is a tuple,

$$P_{TE} = < D_i, F_{trans}, (D_o \cup V), T_{trans} >$$

where

- D_i is the input dataset for the transformation event,
- D_o is the dataset that is the result of the transformation event,
- V is a view or facet that is a result of the transformation event,
- $D_i \cup D_o \cup V \subseteq \mathbb{D}$
- F_{trans} is the transformation function applied to D_i (applied element-wise), and
- T_{trans} is the time the transformation took place.

Definition 3: Run. A **run** can be informally defined as a function from time to dataset(s) and the transformation applied to those dataset(s). Intuitively, a run is a description of how $\mathfrak{G}$ has evolved over time.

So, a run over $\mathfrak{G}$ is a function, $\mathcal{P} : t \rightarrow < D, f >$ where t is an element in the time domain, D is the state of all datasets and views in $\mathfrak{G}$, and f is the function applied at time, t.

A system $\mathcal{R}$ over $\mathfrak{G}$, i.e. $\mathcal{R}(\mathfrak{G})$, is a set of all runs over $\mathfrak{G}$. We say that $< \mathcal{P}, t >$ is a point in system $\mathcal{R}$ if $\mathcal{P} \in \mathcal{R}$.

$\mathcal{P}$ captures our notion of"prospective provenance".

Definition 4: Trace. Let $\alpha =< \mathcal{P}, t > \in \mathcal{R}(\mathfrak{G})$. The trace of α, denoted by, $\overrightarrow{\alpha}$, is the sequence of pairs $< r_i, t_i >$ where r_i is the $i-$th run at time t_i. The set of all traces of $\mathfrak{G}$, $Tr(\mathfrak{G})$, is the set $\{ \overrightarrow{\alpha} \mid \alpha \in \mathcal{R}(\mathfrak{G}) \}$. An element of $Tr(\mathfrak{G})$ is a trace of $\mathfrak{G}$. A trace captures our notion of retrospective provenance.

3 Provenance-Based Data Transformation

3.1 Provenance Model

Community efforts towards the convergence into a common provenance model led to the Open Provenance Model (OPM)[2] OPM descriptions allow interoperability on the level of workflow structure. This model allows systems with different provenance representations to share at least a workflow-level semantics. OPM, however, is not intended to be a complete provenance model, demanding the complementary use of additional provenance models in order to enable uses of provenance which requires higher level of semantic interoperability. This work targets interoperable provenance representations of IDT and ETL workflows using a three-layered approach to represent provenance. In this representation, the bottom layer represents the basic workflow semantics and structure provided by OPM, the second layer extends the workflow structure provided by OPM with Cogs [6], a provenance vocabulary that provides a rich type structure for describing ETL transformations in the provenance workflow, and voidp

[2] http://openprovenance.org/.

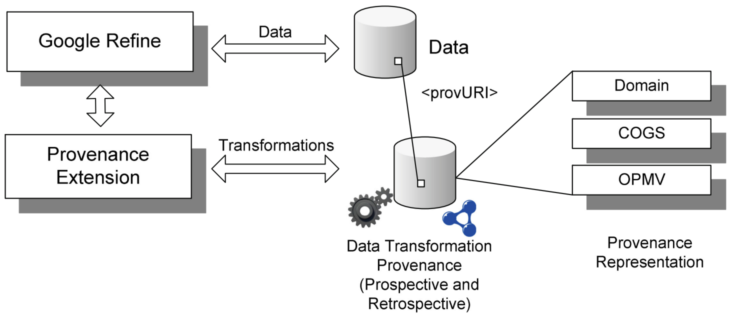

Google Refine
Data
Data
<provURI>
Domain
COGS
OPMV
Provenance Extension
Transformations
Data Transformation Provenance (Prospective and Retrospective)
Provenance Representation

Fig. 2. The system architecture as applied to Google Refine

[8], a provenance extension for the void[3] vocabulary, that allows data publishers to specify the provenance relationships of the elements of their datasets. The third layer consists of a domain specific schema-level information of the source and target datasets or classes/instances pointing to specific elements in the ETL process. In our implementation, the third layer contains the mapping of instances to source code elements. Fig. 2 shows how we applied the model (and architecture) to Google Refine, a popular IDT platform.

3.2 Provenance Capture

There are two major approaches for representing provenance information, and these representations have implications on their cost of recording. These two approaches are: (a) The (Manual) Annotation method: Metadata of the derivation history of a data are collected as annotation. Here, provenance is pre-computed and readily usable as metadata, and (b) The Inversion method: This uses the relationships between the input data, the process used to transform and to derive the output data, giving the records of this trace.

The Annotation method is coarser-grained and more suitable for slowly-changing transformation procedures. For more highly-dynamic and time-sensitive transformations, such as IDT procedures, the Inversion method is more suitable. We map the provenance data to the three-layered data transformation model provenance model as described in Sect. 3.1.

After choosing the representation mechanism, the next questions to ask are: (a) what data transformation points would generate the provenance data salient to our provenance needs, and (b) what is the minimal unit of a dataset to attach provenance to. For our system, we choose an Interactive Data Transformation Event to capture these two questions and this is made up of the data object being transformed, the transformation operation being applied to the data, the data output as a result of the transformation, and the time of the operation (as stated in Sect. 2.3).

[3] http://vocab.deri.ie/void/guide.

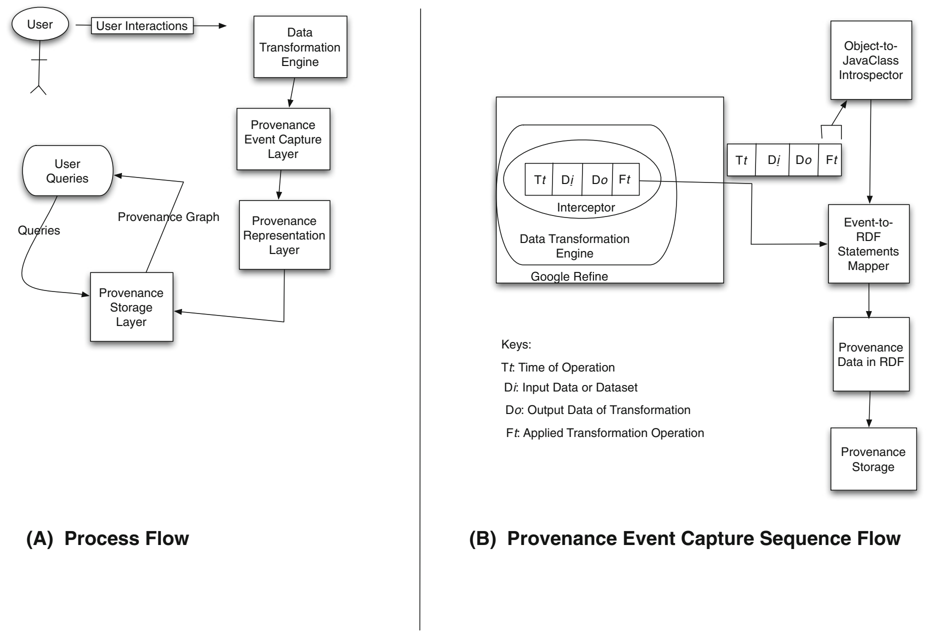

User
User Interactions
Data Transformation Engine
Provenance Event Capture Layer
User Queries
Provenance Graph
Queries
Provenance Representation Layer
Provenance Storage Layer
(A) Process Flow
Object-to-JavaClass Introspector
Tt
Di
Do
Ft
Interceptor
Data Transformation Engine
Google Refine
Event-to-RDF Statements Mapper
Provenance Data in RDF
Provenance Storage
Keys:
Tt: Time of Operation
Di: Input Data or Dataset
Do: Output Data of Transformation
Ft: Applied Transformation Operation
(B) Provenance Event Capture Sequence Flow

Fig. 3. Process and provenance event capture sequence flows

3.3 Process Flow

Fig. 3(A) depicts the process flow that we adopted, and it consists of the following:

1. The Provenance Event Capture Layer, which consists of the following layers and operations' sequence (Fig. 3(B)): (i) The Interceptor Layer: Here, user interactions are intercepted and provenance events extracted from these interactions; (ii) Object-to-JavaClass Introspector: The inputs to this layer are the Transformation Operation chosen to transform the data. We employ the Java language reflection mechanism to elicit the full class path of the operation performed. Eliciting the full class path is useful for the following reasons: (a) It allows us to have a pointer to the binary of the programs doing the transformation, and (b) This pointer to the program binary allows the connection between the full semantics of the program and the data layer. The outputs of this layer are the full class paths of the transformation operations; (iii) Event-to-RDF Statements Mapper: it receives the provenance event and is responsible for the generation of the RDF predicates.
2. These events are then sent to the **Provenance Representation Layer**, which encodes the captured events into RDF using the provenance representation described in Sect. 3.1.
3. These events, represented in RDF, are then sent to the **Provenance Storage Layer**, which stores them in its Knowledge Base (KB).

4 Google Refine: An Exemplar Data Transformation/Curation System

Google Refine[4] (GRefine) is an exemplar Interactive Data Transformation system, and some of its mechanisms include: (i) ability to import data into GRefine from different formats including tsv, csv, xls, xml, json, and google spreadsheets; (ii) GRefine supports faceted browsing[5], such as: Text Facets, Numeric Facets, and Text Filters; (iii) Editing Cells, Columns, and Rows, using GRefine's editing functions; and (iv) The provision of an extension framework API.

4.1 Making Google Refine Provenance-Aware

Some of the design decisions that must be made when making an application provenance-aware is to ask "what" type of provenance data to capture, "when" to collect the said provenance data, and "what" type of method to use for the capture.

As regards to "when" to collect provenance data: (a) provenance data can be collected in real-time, i.e. while the workflow application is running and the input dataset(s) are being processed and used, or (b) ex-post (after-the-fact), i.e. provenance data is gathered after a series of processing events or a sequence of activities has completed.

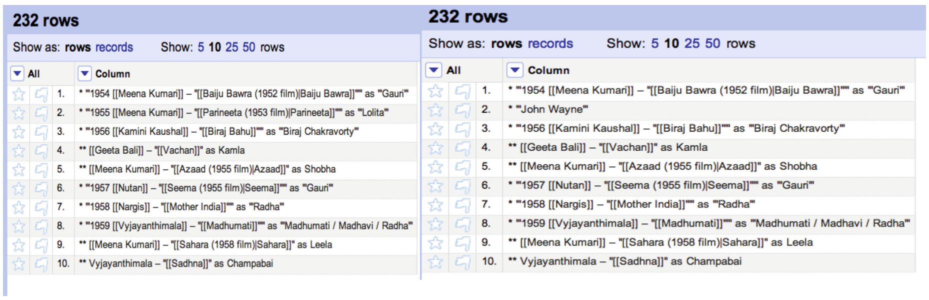
As regards to "what" type of method to use for collection, provenance data collection methods fall into three types: (a) Through "User annotation": A human data entry activity where users enter textual annotations, capturing, and describing the data transformation process. User-centered metadata is often incomplete and inconsistent [10]. This approach imposes a low burden on the application, but a high burden on the humans responsible for annotation; (b) An automated provenance instrumentation tool can be provided that is inserted into the workflow application to collect provenance data. This places a low burden on the user, but a higher burden on the application in terms of process cycles and/or memory, and (c) Hybrid method: This method uses an existing mechanism, such as a logging or an auditing tool, within the workflow application to collect provenance data.

We built an automated instrumentation tool to capture provenance data from user operations as they use the system (Fig. 2). This approach incurs very little burden on the user.

4.2 Usage Scenarios and Data Transformation Experimentation Using Google Refine

Here, we describe how we have used GRefine to capture data transformation provenance events, how these events have been represented using our provenance models (described in Sect. 3.1), and how we have made use of the IDT algebra

[4] http://code.google.com/p/google-refine/.
[5] http://code.google.com/p/google-refine/wiki/FacetedBrowsingArchitecture.



232 rows

Show as: **rows** records Show: 5 **10** 25 50 rows

	All	Column
☆	1.	* '''1954 [[Meena Kumari]] – "[[Baiju Bawra (1952 film)|Baiju Bawra]]"'' as '''Gauri'''
☆	2.	* '''1955 [[Meena Kumari]] – "[[Parineeta (1953 film)|Parineeta]]"'' as '''Lolita'''
☆	3.	* '''1956 [[Kamini Kaushal]] – "[[Biraj Bahu]]"'' as '''Biraj Chakravorty'''
☆	4.	** [[Geeta Bali]] – "[[Vachan]]" as Kamla
☆	5.	** [[Meena Kumari]] – "[[Azaad (1955 film)|Azaad]]" as Shobha
☆	6.	* '''1957 [[Nutan]] – "[[Seema (1955 film)|Seema]]"'' as '''Gauri'''
☆	7.	* '''1958 [[Nargis]] – "[[Mother India]]"'' as '''Radha'''
☆	8.	* '''1959 [[Vyjayanthimala]] – "[[Madhumati]]"'' as '''Madhumati / Madhavi / Radha'''
☆	9.	** [[Meena Kumari]] – "[[Sahara (1958 film)|Sahara]]" as Leela
☆	10.	** Vyjayanthimala – "[[Sadhna]]" as Champabai

232 rows

Show as: **rows** records Show: 5 **10** 25 50 rows

	All	Column
☆	1.	* '''1954 [[Meena Kumari]] – "[[Baiju Bawra (1952 film)|Baiju Bawra]]"'' as '''Gauri'''
☆	2.	* '''John Wayne'''
☆	3.	* '''1956 [[Kamini Kaushal]] – "[[Biraj Bahu]]"'' as '''Biraj Chakravorty'''
☆	4.	** [[Geeta Bali]] – "[[Vachan]]" as Kamla
☆	5.	** [[Meena Kumari]] – "[[Azaad (1955 film)|Azaad]]" as Shobha
☆	6.	* '''1957 [[Nutan]] – "[[Seema (1955 film)|Seema]]"'' as '''Gauri'''
☆	7.	* '''1958 [[Nargis]] – "[[Mother India]]"'' as '''Radha'''
☆	8.	* '''1959 [[Vyjayanthimala]] – "[[Madhumati]]"'' as '''Madhumati / Madhavi / Radha'''
☆	9.	** [[Meena Kumari]] – "[[Sahara (1958 film)|Sahara]]" as Leela
☆	10.	** Vyjayanthimala – "[[Sadhna]]" as Champabai

Fig. 4. Google Refine edit scenario (Before and After)

(in Sect. 2.3). We converted the contents of FilmAwardsForBestActress[6] into a GRefine project called “actresses”.

We have applied our definition of a **Run** (as stated in Sect. 2.3) to actual implementations in the Usage Scenarios described below. Also here, we see the Google Refine system as an implementation of an IDT, $\mathfrak{G}$ (from Definition 1 of Sect. 2.3).

Edit Usage Scenario. If we want to change the entry in row 2 (of Fig. 4) from *”’1955 [[Meena Kumari]] ”[[Parineeta (1953 film)—Parineeta]]””’ as ”’Lolita” to “1935 John Wayne” and would like our system to keep a provenance record of this transaction, we can achieve that, in GRefine, by viewing the column as a “Text Facet” and applying the GRefine’s “Edit” operation on row 2. Figure 4 shows us the before and after pictures of our Edit operation.
The **Events-to-RDF Statements Mapper** automatically generates the RDF statements using the following mechanisms. A transformation function, e.g. “Edit”, from GRefine is automatically mapped to a type (an “`rdf:type`”) of `opmv:Process`. What the operation gets mapped to in the Cogs ontology depends on the attribute of the data item that was the domain of the operation. For example, if the data item attribute is a Column, this gets mapped to a Cogs “ColumnOperation” class, while if the item attribute is a Row, this gets mapped to a Cogs “RowOperation” class. Since this is a transformation process, we made use of “`cogs:TransformationProcess`” class. voidp has a single class of ProvenanceEvent and every transformation operation is mapped to “`voidp:ProvenanceEvent`”.

The system’s Object-to-JavaClass Introspector is used to elicit the actual Java class responsible for the transformation. We have made use of a Cogs property, “`cogs:programUsed`”, to specify the full path to this Java class. It allows the generated provenance data to communicate with the program semantics, in this way the intensional program semantics is linked up with the provenance extensional semantics. The data item that is actually made use of in the transformation

[6] http://en.wikipedia.org/w/index.php?title=Filmfare_Award_for_Best_Actress&action=edit&section=3.

process is mapped to "`opmv:Artifact`". To give us the process flow that is part of the transformation, we used two opmv properties: (a) "`opmv:wasDerivedFrom`", which tells us from which artifact this present data item is derived from, and (b) "`opmv:wasGeneratedBy`", which tells us from which process this data item is generated from. In order to store temporal information of when the derivation took place, we used "`opmv:wasGeneratedAt`", an opmv property that tells us the time of transformation.

The result of the transformation operation as RDF statements is below:

```
@prefix id: <http://127.0.0.1:3333/project/1402144365904/> .
id:MassCellChange-1092380975 rdf:type opmv:Process,
cogs:ColumnOperation, cogs:TransformationProcess, voidp:ProvenanceEvent ;
opmv:used <http://127.0.0.1:3333/project/1402144365904/
                                        MassCellChange-1092380975/1_0> ;
cogs:operationName"MassCellChange"^^xsd:string;
cogs:programUsed "com.google.refine.operations.cell.
                                MassEditOperation"^^xsd:string;
rdfs:label  "Mass edit 1 cells in column ==List of winners=="^^xsd:string.

<http://127.0.0.1:3333/project/1402144365904/MassCellChange-1092380975/1_0>
rdf:type opmv:Artifact ;
rdfs:label  "*'''1955 [[Meena Kumari]]
     '[[Parineeta (1953 film)|Parineeta]]''''' as'''Lolita'''"^^xsd:string.

http://127.0.0.1:3333/project/1402144365904/MassCellChange-1092380975/1_1>
rdf:type    opmv:Artifact ;
rdfs:label  "*'''John Wayne'''"^^xsd:string;
opmv:wasDerivedFrom <http://127.0.0.1:3333/project/1402144365904/
                  MassCellChange-1092380975/1_0>;
opmv:wasGeneratedBy <http://127.0.0.1:3333/project/1402144365904/
                 MassCellChange-1092380975>;
opmv:wasGeneratedAt "2011-11-16T11:2:14"^xsd:dateTime.
```

Row Removal Usage Scenario. Let us say we want to remove the rows that have lines starting with two equal signs such as this: "==". One way to achieve this in GRefine is to do the following: "Column → Text Filter → == → All → Edit rows → Remove all matching rows".

Our operation removed seven rows, and the places where they were removed are also shown by the RDF statements, in `rdfs:label`, below:

```
id:RowRemovalChange-1030462081
rdf:type    opmv:Process, cogs:RowOperation,
                  cogs:TransformationProcess, voidp:ProvenanceEvent ;
opmv:used   <http://127.0.0.1:3333/project/1402144365904/
                                 RowRemovalChange-1030462081/1_0> ;
cogs:operationName  "RowRemovalChange"^^xsd:string;
cogs:programUsed        "com.google.refine.operations.row.
                                 RowRemovalOperation"^^xsd:string;
rdfs:label  "Remove 7 rows"^^xsd:string.

<http://127.0.0.1:3333/project/1402144365904/
                                       RowRemovalChange-1030462081/1_0>
rdf:type    opmv:Artifact ;
```

```
rdfs:label  ""^^xsd:string.

http://127.0.0.1:3333/project/1402144365904/
                         RowRemovalChange-1030462081/1_1>
rdf:type    opmv:Artifact ;
rdfs:label  "[0, 11, 42, 84, 126, 174, 227]"^^xsd:string;
opmv:wasDerivedFrom <http://127.0.0.1:3333/project/1402144365904/
                                                    RowRemovalChange-1030462081/1_0>;
opmv:wasGeneratedBy <http://127.0.0.1:3333/project/1402144365904/
                                                    RowRemovalChange-1030462081>;
opmv:wasGeneratedAt "2011-11-16:11:4:58"^^xsd:dateTime.
```

In the next section, we will describe the requirements we used for analysis.

5 Analysis and Discussion

Our approach and system were evaluated in relation to the set of requirements listed in [3,6] and enumerated below:

1. **Decentralization:** deployable one database at a time, without requiring co-operation among all databases at once. Coverage: High. **Justification:** Use of Semantic Web Standards and vocabularies (OPMV + Cogs + voidp) to reach a decentralized/interoperable provenance solution.
2. **Data model independency:** should work for data stored in flat file, relational, XML, file system, Web site, etc., model. Coverage: High. **Justification:** Minimization of the interaction between the data level and the provenance level. Data is connected with its provenance descriptor by a provenance URI and the provenance representation is normalized as RDF/S.
3. **Minimum impact to existing IDT practice:** Provenance tracking is invisible to the user. Coverage: High. **Justification:** The provenance capture is transparent to the user. Provenance is captured by a lightweight instrumentation of the IDT platform, mapping program structures to the provenance model.
4. **Scalability:** to situations in which many databases cooperate to maintain provenance chain. Coverage: High. **Justification:** Usage of Semantic Web standards and vocabularies for provenance representation allows for the co-operation of multiple platforms for provenance management.

6 Related Work

We categorize related work into three: (i) provenance management for manually curated data [3], (ii) interactive data transformation models and tools [4,5], and (iii) data transformation models for databases [7].

Buneman et al. [3] propose a model for recording provenance of data in a single manually curated database. In their approach, the data is copied from external data sources or modified within the target database creating a copy-paste model for describing user actions in assimilating external datasources into curated database records.

Raman and Hellerstein [5] describe Potters Wheel, an interactive data cleaning system which allowed users to specify transformations through graphic elements. Similarly, Wrangler [4] is an interactive tool based on the visual specification of data transformations. Both systems are similar to Google Refine: [5], however, provides a more principled analysis of the interactive data transformation process, while [4] focuses on new techniques for specifying data transformations. The IDT and provenance model proposed in our work can be directly applied to both systems.

Davidson [7] analyses the requirements for the construction of a formalism for modeling the semantics of database transformations and propose a declarative language for specifying and implementing these transformations and other constraints. They conclude that approaches for modeling database transformations based on a fixed-set of well-defined transformations can inherently limit the expressivity of the transformation model. An additional conclusion is that a high-level language is necessary to express general transformations but that such language should have a well-defined formal semantics.

Our work is different from that of Davidson [7]. Our solution provides a best-effort and pay-as-you-go semantics for the data transformation, focusing on an improvement of the level of semantic interoperability across data provenance of data transformation representations. We also provide a formal model of the transformation system, on which a language can be constructed to generate such a data transformation system.

7 Conclusion

The world contains an unimaginably vast amount of data which is getting ever vaster ever more rapidly. These opportunities demand data transformation efforts for data analysis and re-purposing. Interactive data transformation (IDT) tools are becoming easily available to lower barriers to data transformation challenges. Some of these challenges will be solved by developing mechanisms useful for capturing the process flow and data lineage of these transformation processes in an interoperable manner. In this paper, we provide a formal model of IDT, a description of the design and architecture of an ontology-based provenance system used for mapping data transformation operations, and its implementation and validation on a popular IDT platform, Google Refine. We shall be making our provenance extension to Google Refine publicly accessible very soon.

Acknowledgements. This work was supported by the EnAKTing project, funded by EPSRC project number EP/G008493/1 and by the Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (Lion-2).

References

1. Buneman, P.: Curated databases. In: Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data (SIGMOD) (2006)

2. Curry, E., Freitas, A., O'Riain, S.: The role of community-driven data curation for enterprises. In: Wood, D. (ed.) Linking Enterprise Data, pp. 25–47. Springer, Boston (2010)
3. Buneman, P., Chapman, A., Cheney, J., Vansummeren, S.: A provenance model for manually curated data. In: Moreau, L., Foster, I. (eds.) IPAW 2006. LNCS, vol. 4145, pp. 162–170. Springer, Heidelberg (2006)
4. Kandel, S., Paepcke, A., Hellerstein, J., Heer, J.: Wrangler: interactive visual specification of data transformation scripts. In: ACM Human Factors in Computing Systems (CHI) (2011)
5. Raman, V., Hellerstein, J.: Potter's wheel: an interactive data cleaning system. In: Proceedings of the 27th International Conference on Very Large Data Bases (2001)
6. Freitas, A., Kämpgen, B., Oliveira, J.G., O'Riain, S., Curry, E.: Representing interoperable provenance descriptions for ETL workflows. In: Proceedings of the 3rd International Workshop on Role of Semantic Web in Provenance Management (SWPM 2012), Extended Semantic Web Conference (ESWC), Heraklion, Crete (2012)
7. Davidson, S., Kosky, A., Buneman, P.: Semantics of database transformations. In: Thalheim, B., Libkin, L. (eds.) Semantics in Databases 1995. LNCS, vol. 1358, pp. 55–91. Springer, Heidelberg (1998)
8. Omitola, T., Zuo, L., Gutteridge, C., Millard, I., Glaser, H., Gibbins, N., Shadbolt, N.: Tracing the provenance of linked data using voiD. In: The International Conference on Web Intelligence, Mining and Semantics (WIMS 2011) (2011)
9. Deelman, E., Gannon, D., Shields, M., Taylor, I.: Workflows and e-Science: an overview of workflow system features and capabilities. Future Gener. Comput. Syst. **25**(5), 528–540 (2009)
10. Newhouse, S., Schopf, J.M., Richards, A., Atkinson, M.: Study of user priorities for e-Infrastructure for e-Research (SUPER). In: UK e-Science Technical report Series Report UKeS-2007-01 (2007)