# Provenance, Lineage, and Workflows

Sidra Islam
sidra@cs.brown.edu

*Computer Science Department, Brown University, RI, USA*
Advised by Stanley B. Zdonik

May 3, 2010

## Abstract

In Computer Science, Provenance - also known as lineage and pedigree - describe the source and derivation of data. Data provenance is key to the management of scientific data and has recently been recognized as central to the trust one places in data. This paper focus attention on the importance and difficulty of provenance tracking in practice. We discuss a taxonomy of data provenance characteristics and focus primarily on scientific workflow approaches.

## 1 Introduction

Information describing the origin, derivation, history, custody, or context of an object, generally referred to as *provenance* (also referred to as *lineage* and *pedigree*). Provenance is important to manage scientific data and to understand the authenticity, integrity and trustworthiness of the information about digital objects. From a scientific point of view, data sets are useless without knowing the exact provenance and processing pipeline used to produce derived data sets. We will discuss system-development challenges of tracking information efficiently as well as theoretical challenge of determining what exactly should be tracked in order to accomplish a particular task.

There are two general approaches that have been used in scientific computation research in recording provenance; *workflow* or *coarse-grain* provenance, and *dataflow* or *fine-grain* provenance.

Information describing how derived data has been calculated from raw observations is referred to as "Coarse-grain" or "Workflow" provenance. Widespread use of workflow tools for processing scientific data facilitate capturing provenance information. The workflow process describes all the steps involved in producing a given data set and, hence, captures its provenance information.

Information describing how data has moved through a network of databases is referred to as "Fine-grain" or "Dataflow" provenance.

Fine-grain provenance is further categorized into; where-, how-, and why-provenance.[12] A query execution simply copy data elements from some source to some target database and where-provenance identifies these source elements where the data in the target is copied from. Why-provenance provides justification for the data elements appearing in the output and how-provenance describes how some parts of the input influenced certain parts of the output. Consider the following SQL query over two relations Student(id, name, major, deptid) and Dept(deptid, dname).

```
select Student.name, Student.major, Dept.dname
from Student, Dept
where Student.deptid = Dept.deptid
```

Assume that after executing above query, we get result (Sidra, CS). The where-provenance of "Sidra" is the name attribute of some Student tuple whose value is "Sidra". There can be more than one Student tuple whose value of the name attribute is "Sidra" so, why-provenance of (Sidra, CS) can make this precise selection using *where* clause of the above query where a tuple from the Student table and a tuple from the Dept table agree on the deptid value. Similarly, how-provenance of (Sidra, CS) describes how this particular tuple from the Student and Dept table is selected with the help of *where* clause of the query and ignoring rest of the tuples in the Student and Dept tables.

## 2 Desired Properties

In this section, we discuss requirements of what make good provenance recording a reality.

- A good provenance record should provide an adequate description of which external data sources have been copied or consulted and how further reorganized or corrected. Provenance of data should also explain when changes have been made, who is responsible for them, and how database has evolved over time.
- Automated provenance recording support is desirable because manually recorded provenance of data can easily be incomplete or incorrect.
- If errors are discovered, the provenance record should make it possible to identify other copies that may require correction.
- If provenance recording behavior of a system is inconsistent then it gives a false sense of security that satisfactory provenance information is being recorded when it is not. To be trustworthy, a system should provide transparent, high-level guarantees concerning the provenance it records.

Provenance has significant implications on privacy, anonymity, and other areas of security that are of current interest. So, a provenance-aware security model is needed to provide integrity for data which is considered a challenging task to achieve in security.

- Some forms of provenance records should be able to ensure repeatability or reproducibility of scientific experiments to avoid duplication of efforts.

## 3 Life Cycle of Provenance

We summarize key principles of provenance in the form of provenance life cycle. As shown in Figure 1, provenance-aware applications create documentation of execution which describes what actually occurred at execution time. A long-term persistent, secure storage referred as provenance store is used to store this process documentation. Provenance store and its contents might need to be managed, maintained, or curated and can be queried to retrieve and analyze provenance information. Provenance queries over process documentation are used to help user to find information of interest. A provenance query must be able to identify data of interest with respect to a given characterized, documented event.

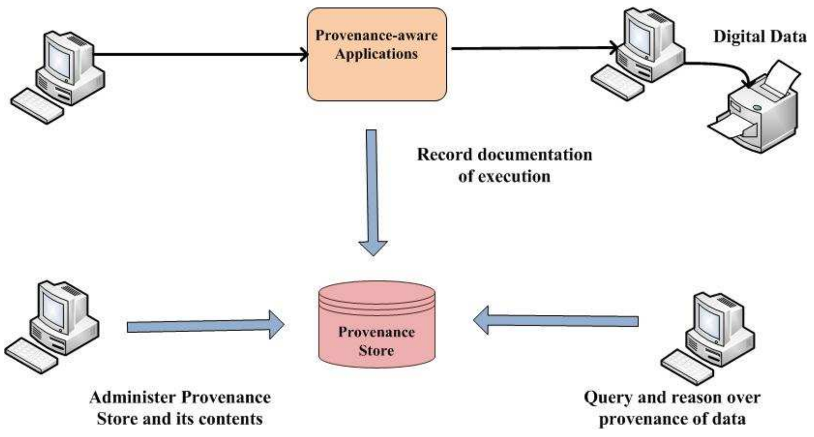
Provenance-aware Applications
Digital Data
Record documentation of execution
Provenance Store
Administer Provenance Store and its contents
Query and reason over provenance of data

Figure 1: Provenance Life Cycle

Documentation of execution for many provenance-aware applications must be interleaved continuously with execution in order to help designers to be able to distinguish a specific item documenting part of a process from whole process of documentation[9].

To describe how provenance-aware applications are executed, Figure 2 shows an open data model with the help of p-assertions which are assertions made by an individual application service involved in the process. Interaction p-assertions help to analyze an execution and verify its validity or compare it with other executions. Therefore, documentation of execution includes interaction p-assertions, or descriptions of the contents of an input, output by a computational service that has sent or received it. Computational services also provide information in the form of relationship p-assertions which describe how it obtained output data sent in an interaction by applying some function or algorithm to input data from other interactions. In Figure 2, output $O_1$ was obtained by applying function $F_1$ to input $I_1$. Interaction and relationship p-assertions, both give explicit description of the flow of data between and within services in a process. Finally, service state p-assertion is documentation which describes nonfunctional characteristics of execution such as performance or accuracy of services and nature of the computed results. Service state p-assertion in Figure 2 shows internal state of the service in the context of a specific interaction where $t$ is time when an action occurred and $x$ is algorithm used to compute output. It may also include amount of disk and CPU times used by a service in a computations, precise results and application-specific state descriptions.

It is a crucial task to have process documentation structured according to a shared data model for provenance-aware applications to be inter-operable .This proposed open model produce process documentation by application service autonomously in a form which can be used by provenance queries to retrieve information.

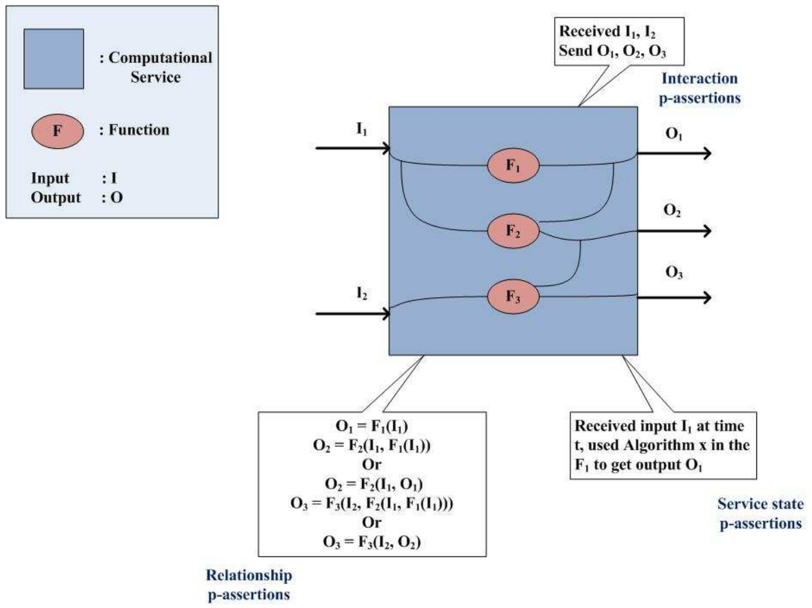

Figure 2: Categories of p-assertions made by a computational service

## 3.1 Use Case (Health Care Management Application)

Figure 3 illustrates the vision of provenance-aware applications through a concrete example of Cancer information Management System (CIMS). CIMS consists of a complex process involving the surgery itself, along with such activities as data collection, requests of Lab reports, notifications and justification reports in order to make decision based on this analyzed information. This provenance-aware application allows medical personnel to find all doctors, lab reports, and patient's local files involved in a decision. Such functionality also can be made available to regulators and patient's families to analyze and understand how particular actions are taken[9].

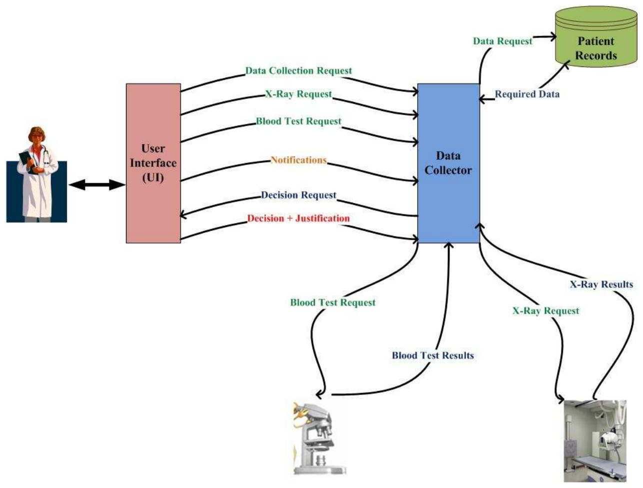

Figure 3: Cancer Information Management System (CIMS)

Figure 4 captures causal and functional data dependencies in execution in the form of directed acyclic graph (DAG). Data-flow DAG indicates how specific data item is produced, used, and becomes an important element of provenance representation.

CIMS example includes only a limited number of components, but real-life examples involve vast amount of documentation, users, doctors, patients, or regulatory authorities, taking benefit from a powerful and accurate provenance-query facility.

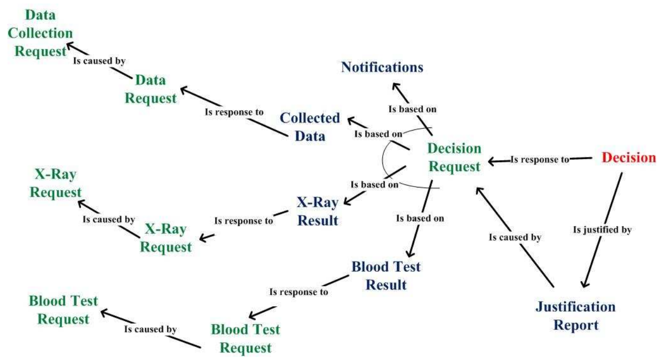

Figure 4: Provenance directed acyclic graph of CIMS

## 4 Taxonomy of Provenance

Taxonomy of data provenance categorizes provenance system based on why they record provenance, what they describe, how they represent and store provenance, and different ways to retrieve information from it. Figure 5 summarizes taxonomy of provenance techniques that have been taken to support data provenance requirements for specific domains. Following is briefly discussion about each of its aspect[8].

### 4.1 Application of Provenance

The importance of recording provenance has recently sparked interest in database research and have several applications as follows[6]:

- **Attribution**: Provenance information help to establish the historical importance of a work, to determine the legitimacy of current ownership and authenticity of a work and also determine liability in case of erroneous data.
- **Annotation**: Process of adding information to existing data is referred to as annotation of data. In some applications, annotations to be propagated from source to output in a systematic way is desirable and a program is used to capture provenance. In some cases, to have annotations propagated from output to source is also important. Annotation propagation is based on provenance.
- **Trust Conditions**: In scientific databases, provenance is important because researchers may have trust conditions for data based on this information.
- **Data Quality**: Provenance records are particularly important that summarize the construction process of scientific databases to assess data quality and data reliability.

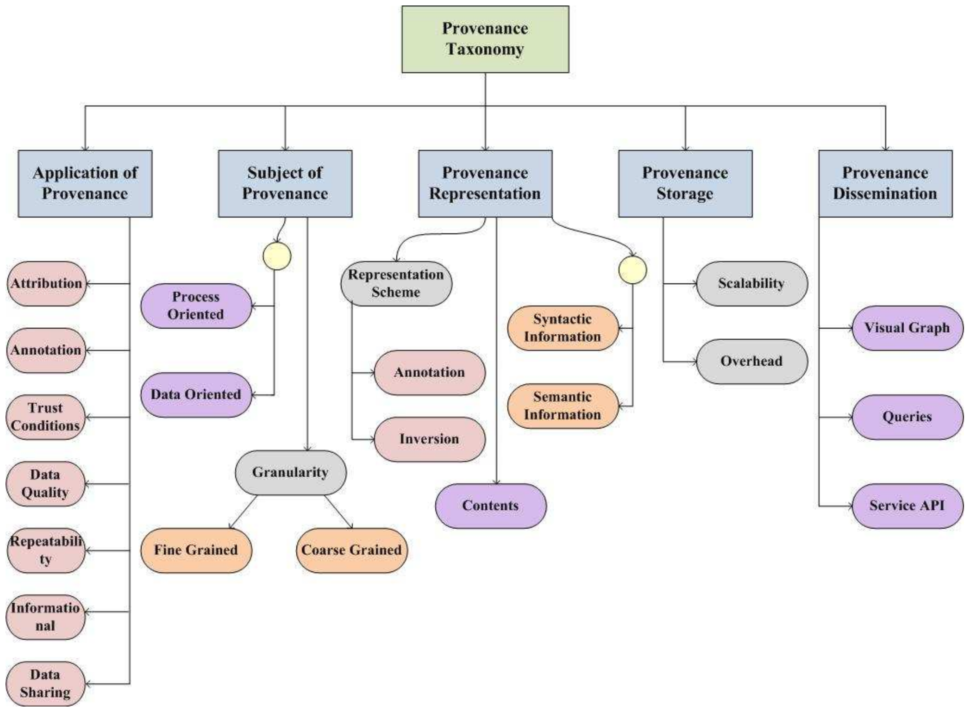

Figure 5: Taxonomy of Provenance

- **Repeatability**: Many scientific data sets are the result of complex analyses or simulations. It is very important to keep a complete record of how the computation was performed in order to avoid duplication of expensive computations, to recover the source data from the output data, to catalog the result and to ensure repeatability.
- **Informational**: Provenance can be queried to provide a context to interpret data and also help to discover data.
- **Data Sharing**: Provenance is central to trust and in data exchange or integration scenario provenance has also been used to describe relationships between source and target data to understand and debug the specification of the integration system.

Provenance is also important to understand the transport of annotation of data elements in scientific databases, to data integration, to probabilistic databases and to keep the record of update and maintenance.

We examine some other applications of provenance models and provenance recording.

**Scientific workflow systems** maintain provenance in order to ensure repeatability, to avoid expensive re-computations, and to asses data quality of scientific databases. Also, recording the change history of a database as a provenance information is considered essential to determine the scientific value of scientific databases.

In **Semantic Web systems**, provenance information in the form of proofs or explanations, help users to understand the meaning of results of inference-based search.

In **Data warehouses**, recording provenance help to trace information in a view to relevant source data in the underlying databases.

**Curated Databases** are scientific databases that are constructed by copying, correcting, annotating, and integrating data from other data sources. Such curated databases should be of high quality because they are becoming a form of recognized scientific publication. Provenance is an essential component to asses the data quality of these manually curated databases and space overhead for doing so is acceptable.

**Probabilistic and uncertain databases** use application of provenance to capture the set of possible instances correctly and to determine whether the sources of tuples are independent in the result of a probabilistic query. This research was done as a part of Trio project, where goal is to manage data, its provenance and uncertainty as one integrated system.

In addition, related ideas and techniques to record provenance also seem to play a role in other research areas such as programming languages (source locations in debugging and error messages) and software engineering (version control, configuration management).

## 4.2 Subject of Provenance

In data processing system, provenance information can be about different resources and at multiple levels of detail. Recording provenance is specifically about the data product in *data-oriented model* whereas, in *process-oriented model*, provenance data is collected about the deriving processes and input, output data products of these processes.

Provenance granularity helps to determine the usefulness of it in an particular domain. The cost of collecting and storing provenance can be inversely proportional to its granularity which is categorized into *Coarse-Grained* and *Fine-Grained.*

### 4.2.1 Coarse-grain or Workflow Provenance

Scientific communities are contributing to distributed data and computational community infrastructure a.k.a. "the grid" in order to share their data and computational services. Scientists are focusing on the development and use of *scientific work-flows*[3] which involve analytical steps, e.g., database access and querying steps, data analysis and mining steps, and many other steps including computationally intensive jobs on high performance cluster computers. Work-flows can be of many different types, e.g., data-intensive, compute-intensive, analysis-intensive, visualization-intensive, etc.

*Workflow or coarse-grain provenance* is the most general approach to provenance which is used to record a complete history of the derivation of some data set including a record of human interaction with the process, tracking the interaction of programs, and also involvement of external devices such as sensors, cameras and other data collecting equipments. A proper record of workflow provenance helps scientists to repeat and validate scientific experiments systematically.

First we illustrate common requirements of scientific work-flows and then we discuss provenance collection framework of two well-known workflow systems, *Kepler* and *Pegasus*.

#### Requirements of Scientific Work-flows

Common requirements of scientific work-flows are following[1]:

**Seamless access to resources and web services**: Web services provide easy ways to access database and execute services remotely through service calls which is a simple solution. But, complex problems are not solved by web services alone, e.g., web service orchestration and 3rd party transfer[1].

**Service composition and Re-usability**: The problem of *service composition*, i.e., how to compose simple services to perform complex tasks, and how to design platform independent components so that they are easily reusable, are important research areas to consider.

**Scalability**: Interfaces that are suitable to Grid middle-ware components also known as Compute-Grid, and Data-Grid are necessary in order to support data-intensive and compute-intensive work-flows.

**Execution mode for remote server**: Execution mode is required for remote server to run long work-flows without keep establishing connection to a user's client application which has initiated workflow execution.

**Reliability and Fault-tolerance**: Some computational environments are more likely to have flaws and are un-reliable than others. Alternate strategies must be available to make a workflow more fault-tolerant and reliable.

**User-interaction**: User decisions and interactions may be required in many scientific work-flows. It could be challenging when user's client application is not connected with remote server while workflow execution is in process and user interaction is required. User can be reconnect with remote server using notification mechanism in order to make decisions and resume paused execution of the scientific workflow.

**Intelligent workflow execution**: If user interaction change the components of a workflow then intelligent execution mode would only re-run the effected part of the system rather than running complete execution from scratch.

**Intelligent semantic links**: A scientific workflow system should assist workflow design in order to indicate which actor parameters might fit together and which datasets might be the input of which workflow, with the help of semantics of data and actors.

**Data Provenance**: A scientific workflow system should be able to record which specific data sets were fed as input, which tools were used in processing, and all the sequence of applied steps in order to repeat computational experiments and re-run the execution of scientific workflow.

**GUI for visualization**: GUI can provide the ability to visualize the execution of a particular workflow through various databases and sequence of applied transformation steps.

## Kepler: A Scientific Workflow System

Kepler is an actor-oriented approach provides an easy-to-use system for scientists for capturing scientific work-flows[1]. Kepler helps to improve workflow creation and execution process so that scientists can design, execute, monitor, re-run, and communicate analytical steps repeatedly with less effort. This scientific workflow system aims to keep track of all types of provenance: in workflow evolution, data and process provenance and efficient management and usage of collected data.

First we give a brief overview of the Kepler and illustrate its architecture, then explain its generic provenance collection framework, and finally we discuss ”smart” re-run feature of Kepler which use extracted provenance information to perform re-run operation efficiently for slightly modified workflow in Kepler.

## Overview and Architecture

Kepler builds on the top of a mature java-based actor-oriented mechanism, Ptolemy software, which is a set of APIs for heterogeneous hierarchal modeling. This approach facilitates modeling and design of complex systems and thus provides a very promising direction for pressing problems such as web service composition and orchestration.

Ptolemy system builds models based on the composition of existing, independent components (e.g., for data movement, database querying, job scheduling, remote execution etc.) called *actors*. Actors communicate with each other through interfaces called *ports* which are connected to one another via *channels*. In addition to the ports, actors have *parameters* which configure and customize the behavior of the actors. A key property of Ptolemy is the observation of the behavior of simulation models when executed using different computational semantics specified by an object called a *director*.

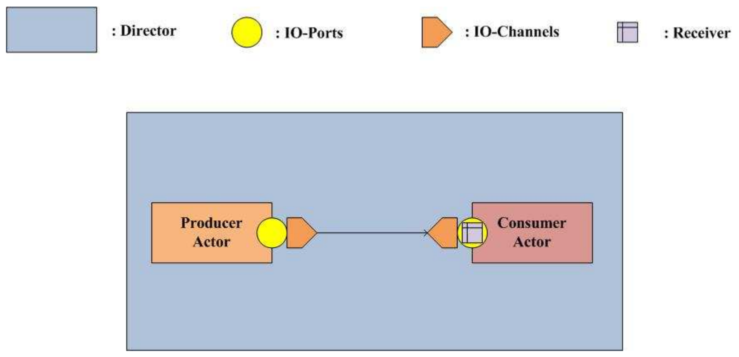

Figure 6: Ptolemy System

Figure 6 shows a producer and consumer actor whose ports are connected with each other via a unidirectional channel. Communication is controlled by a director which can provide an object called *receiver* to mediate the communication between the actors.

The Kepler system can support different types of data- and compute intensive applications ranging from local analytical applications to distributed high-performance and high-throughput pipelines.

Figure 7 shows the architecture of Kepler system where, along with the workflow design and execution feature, much more work needs to be done on a number of built-in system functionalities including support for single sign-in Grid Security Infrastructure(GSI)-based authentication and authorization; creating, publishing, and loading plug-ins as archives using the Vergil graphical user interface(GUI)[2]; semantic annotation of actors, types, and work-flows; and documenting entities of different granularities. One major disadvantage of using Kepler system is that Kepler users cannot develop work-flows in a resource-independent way. Therefore, Kepler system does not provide benefits of workflow portability, optimization, and ease of design.

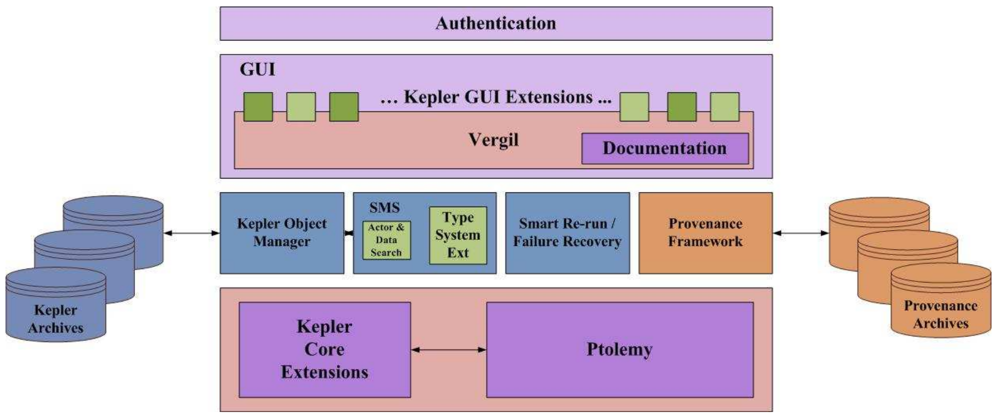

Figure 7: The Kepler System Architecture

**Provenance Framework and Re-run Feature of Kepler**

A highly configurable provenance component is part of the Kepler system that can be easily used with different models of computation using the separation of concerns design principle and is bound visually to the associated workflow. The advantages of this design are: visualization help to see whether provenance is being recorded for a particular run of the workflow, and another advantage is its consistency with the behavior of the Kepler user interface because this provenance framework is according to the Kepler's visual actor-oriented programming paradigm[21].

"Smart" re-run feature of Kepler system enables users to run the workflow again after changing parameter of an actor. Kepler runs only affected part of the workflow after the parameter change, with the help of provenance data and data dependency analysis. Performance of the system using Provenance Recorder (PR) may be greatly affected by the amount of produced and saved provenance information.

## Pegasus: A Scientific Workflow System

Many applications are result of large scientific collaborations, involve large volume of datasets, and many geographically distributed compute and data resources are needed to process them. Pegasus system creates a separation between application development and execution in order to manage the process.

In Pegasus, work-flows that can capture the behavior of the application, are *abstract* at the application-level and only describes the application components and their dependencies. Pegasus takes this abstract work-flows that are described in resource-independent ways and maps them onto appropriate, potentially distributed multiple heterogeneous resources depending on the availability and characteristics of the resources across the wide area networks as shown in the Figure 8.

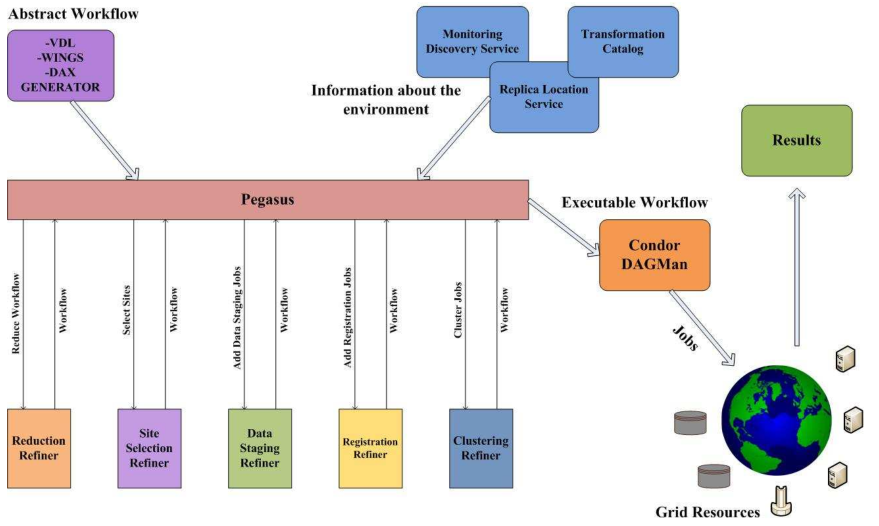

Figure 8: Pegasus converts abstract work-flows into executable work-flows after refinement and then maps them onto the grid resources.

Pegasus use three main ways of specifying the abstract workflow; using a semantic-rich workflow composition tool-Wings, using partial workflow descriptions-via VDL, and by directly specifying the workflow in a XML format DAX (Directed Acyclic Graph in XML). The abstract workflow consists of tasks in the form of logical transformations and logical input output files. Monitoring Discovery Service (MDS) component of the system provides information about the number and type of available resources and their characteristics. Replica

Location Service (RLS) is a distributed replica management system provides information about the data available at the resource and consists of local catalogs that contain information about logical to physical filename mappings and distributed indexes that summarize the local catalog content. Pegasus also queries data registries to find the location of the data referred to in the workflow and queries the Transformation Catalog (TC) to find the location of the workflow component executables[16].

**Pegasus Functionality**: First, Pegasus consults MDS and the pool configuration file to check which resources are available, then next step may modify the structure of the abstract workflow based on the available data products and after that resource selection is performed. Pegasus explicitly perform data transfers since the workflow can be executed across multiple platforms and data need to be staged in and out of the computations. Additionally, where appropriate, intermediate and final data products may be registered and also, Pegasus provides an option to cluster jobs together[14].

After all the resource computation and data selection, Pegasus system creates an executable workflow which can be interpreted by a workflow execution engine Condor DAGMan for execution. DAGMan follows the dependencies described in the executable workflow and releases the tasks to the execution environment. Although, Pegasus allows its users to develop work-flows in a resource-independent way, it does not provide any graphical method of workflow composition and a tool capable of visual workflow execution[2].

**Provenance in Pegasus**: Provenance recording actions take place for each refinement and execution step. Workflow refinement and execution document allows detailed provenance of a data item to be found, e.g., users can find executable workflow steps for each data item and also connection between abstract workflow and executable workflow can be found[13].

Recording copies of all data passing through the system is infeasible. Hence, there is a pressing need to have additional techniques for coping with very large datasets and also for querying provenance data to allow users to find their information of interest easily.

#### 4.2.2 Fine-grain or Dataflow Provenance

In most cases, the whole workflow may not be available or to record workflow provenance may be extremely complicated because "workflow" may be the record of actions on individual components of the database. That is why, *fine-grain provenance* approach is used to derive information of the component of interest which may have simple explanation.

Information describing how data has moved through a network of databases is referred to as "Fine-grain" or "Dataflow" provenance.

## Provenance Management Approach in Curated Databases

In scientific disciplines, curated databases are the result of an investment of a substantial amount of effort by individuals who have organized, interpreted, annotated, corrected, and transfered data from other sources. Mostly, contents of curated databases are derived from other data sources or curated databases. That is why, provenance information describing creation, recording, ownership, processing, or version history of such data is essential and crucial for assessing quality, integrity, and scientific value of data items[5].

To manage provenance in curated databases, focus is on fine-grain provenance because coarse-grain provenance is important in scientific computations, but is not a major concern in curated databases. Provenance information includes local modifications to the database (inserting, deleting, and updating data), global operations such as copying data from external sources, and other user actions involved in constructing a curated database.

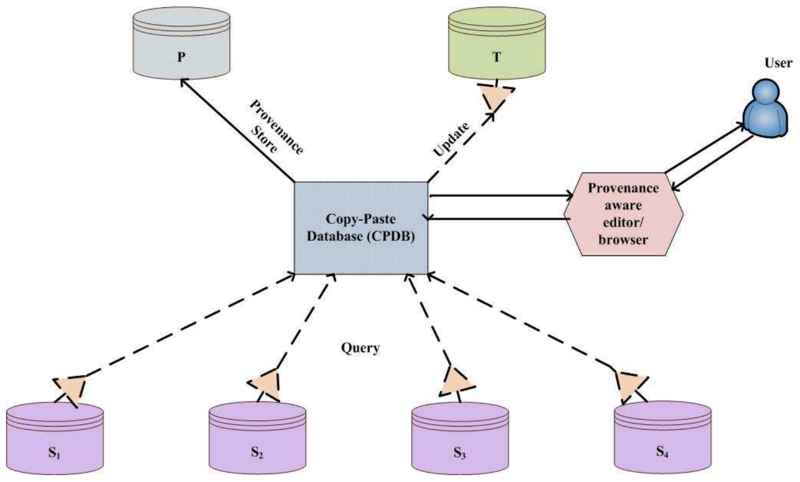

Figure 9: Provenane Architecture

Figure 9 shows proposed approach to track and manage provenance where we assume that all of the user's actions in constructing the target database are captured as a sequence of insert, delete, copy, and paste actions by a provenance-aware application for browsing and editing databases. Local modifications are stored in local database T and provenance links are stored in an auxiliary provenance database P. Provenance links stored in P relate data locations in T with locations in previous versions of T or in external source databases S. Provenance links can be used to review the construction process of target database T and if T is also being archived then these links can provide further details about how different versions of T relate to each other. High-level interfaces that track provenance are essential in order to ensure consistency of target database and its provenance record. In Figure 5, triangles indicate wrappers mapping data sources $S_1$, $S_2$,.....$S_n$, and target database T to an XML view. CPDB "copy-paste database" allows the user to connect to the external source databases, copy source data into the target database, and modify data according to target database's structure. Major concern with this approach is the processing and storage costs which could be un-acceptedly high.

### DBNotes: A Post-It System

DBNotes, a Post-It note system for relational databases where every data item may be associated with zero or more notes or annotations that describe its origin and is propagated with data transparently. The process of annotation propagation is based on provenance where provenance information and flow of data can be traced systematically through tracking back a sequence of transformation steps with the help of annotations[10].

Annotations can also be used to insert some additional information about data items without having to change the underlying schema, e.g., error report could be associated with erroneous data item and will notify other users after propagation to other databases along transformations. This feature of DBNotes system is useful where database schema is often proprietary. Overall, annotations help to asses the quality, and reliability of resulting databases.

### Features of DBNotes System

There are four main features of DBNotes system.

**Propagating Annotations**: DBNotes system has a feature of pSQL, an extension of a fragment of SQL, that allows users to specify how annotations should propagate through a SQL query. pSQL supports three types of propagation schemes.

Annotations can be propagated

- according to where data is copied from.
- according to where data is copied from in all equivalent queries.
- according to the user specifications.

**Querying Annotations**: The second feature of DBNotes system is to use pSQL to query both data and its annotations. For example, user can retrieve all data items derived from a particular source with the help of this feature.

**Explanation of Provenance Data**: DBNotes provides a detailed description on the provenance of an attribute value or annotation obtained in the result of a query transformation.

**Tracing the Provenance and Flow of Data**: Another feature of DBNotes is its ability to visually trace the provenance and flow of a particular data item through various databases and transformation steps.

**System Architecture**

Figure 10 illustrates the architecture of the system. The translator module translates a pSQL query into an SQL query according to DBNotes's underlying storage scheme for annotations. The postprocessor module merges together annotations which have identical origins. The provenance and the flow tracer modules and explainer module are described above.

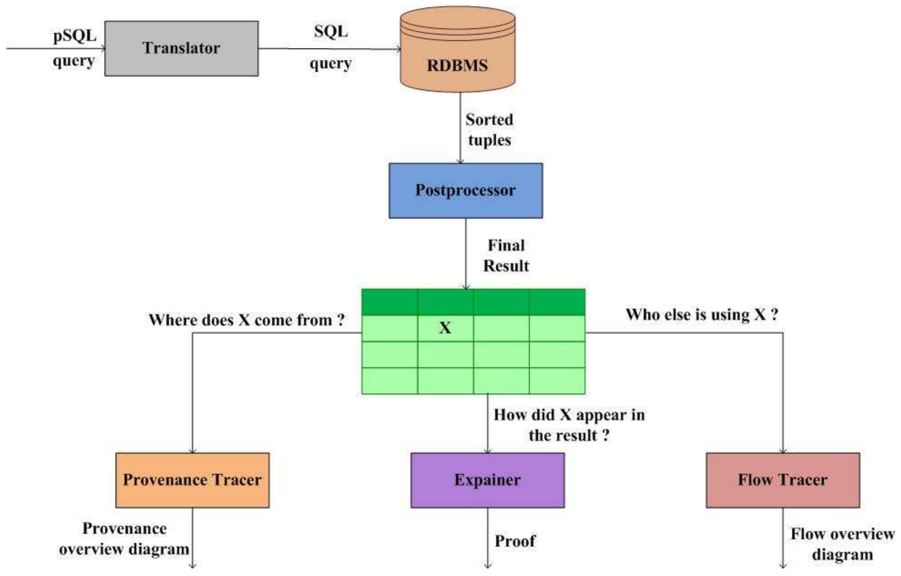

Figure 10: Architecture of DBNotes

DBNotes is the first Post-It notes system for relational databases that allows a user to specify how annotations should propagate, how to query annotations and analyze the provenance information and flow of data through databases generated by query transformations. But this system only supports annotations on attributes of tuples and can be extended to handle annotations on tuples or relations. Also, in DBNotes, annotations' propagation is based on only where-provenance. The performance of this annotation management system needs to be investigated in order to compare it with other storage schemes.

## 4.3 Representation of Provenance

Provenance information can be represented with the help of different techniques, some of which depend on underlying data processing system. Cost of collecting, storing provenance and richness of its usage have influence on the representation of provenance.

Provenance approaches use either annotations or inversion to represent provenance. Information collected as annotations describe the derived history of a data product, source data and processes. On the other hand, inversion method is used to invert some derivation to find the input data supplied to them in order to derive the output data.

It is a challenging task to develop a suitable lineage standard to represent provenance across diverse scientific disciplines. Many provenance systems that use annotations have adopted XML for representing pedigree information. Some systems also capture semantic information within provenance using domain ontologies. All the relationships and concepts used in the provenance can be expressed precisely with the help of ontologies.

## 4.4 Provenance Storage

Provenance information can be a lot larger as compare to data sets it describes. So the manner in which the provenance data is stored is important to its scalability. Annotations method is less scalable than inversion but its storage needs can be reduced by storing only immediate preceding transformation step that creates the data and then derive complete history by inspecting provenance information recursively. Provenance can be tightly coupled to the data and located in the same data storage system which can help maintaining information integrity but it can make lineage information hard to publish and search just provenance.

collecting, storing and managing provenance data is an expensive operation, that is why less frequently used provenance information can be archived to reduce storage overhead.

## 4.5 Provenance Dissemination

Diverse and efficient workflow tools should be used by the system to retrieve useful provenance information. Mostly, derivation graphs that users can browse are used to disseminate provenance.

Users can also query a certain workflow to locate all of the datasets generated by it. Workflow information help to replicate data at another location. If original data sources have modified, still users can re-generate datasets using their derivation history. Unfortunately, there are no efficient ways to store and query workflow based lineage information. Current projects, e.g., Trio and GridDB use recursive queries[4] and recursive tables to represent workflow. Such mechanisms does neither scale for large workflow processes nor for large collections of datasets.

Moreover, service APIs also allow users to retrieve provenance information for their own purposes.

## 5 Projects using Data Provenance Techniques

In this section, we will briefly discuss projects that have done research in this field and are using different techniques to track data provenance.

### 5.1 Chimera

In collaborative environment, Chimera[22] manages the derivation and analysis of data objects and store data derivation steps so that this derivation history of datasets can be used to re-generate derived data.

Chimera uses novel idea based on provenance to plan and estimate the cost of dataset on-demand re-generation with the help of workflow planner which selects optimal plan to allocate resources.

### 5.2 myGrid

myGrid modeled as work-flows in a Grid environment[23] and provides middleware in support of *in silico* (computational laboratory) experiments in biology. myGrid provides services including resource discovery, and provenance management which enable integration and present a semantically enhanced information model for bioinformatics.

The myGrid Information Repository (mIR) data service is a central repository built over a relational database to store provenance information about experimental components which can be used for on-demand knowledge discovery. The process of information retrieval can generate provenance data by executing a workflow and because of this, another workflow can also be executed if previous workflow execution have updated its contents.

### 5.3 CMCS

CMCS is a meta-data based heterogeneous data management project designed to support collaboration in between multiple scientific disciplines[24][25]. Data modifications indicated by provenance meta-data can trigger workflow execution to update dependent datasets.

### 5.4 ESSW

The Earth System Science Workbench (ESSW)[26] is a data storage system and meta-data management for earth science researchers. In ESSW project, provenance information is used to assess quality of data products and detect errors in derived datasets.

### 5.5 Trio

In Trio project, lineage information[27] is traced to view data in a proposed database system which has data provenance and data accuracy as inherent components. A database view can be modeled as a query tree that is evaluated bottom-up, starting with leaf operators having tables as inputs and successive parent operators taking as input the result of a child operator. Trio uses inverse query of the view query that operates on the materialized view, and recursively moves down the query tree to identify the source tables in the leaves that form the view data's lineage.

### 5.6 ORCHESTRA

ORCHESTRA[19], a collaborative data sharing system is a flexible scheme for sharing data among broader communities by allowing loosely coupled confederations of sites, each of which maintains a local schema, and a fully autonomous, edit-able local data instance.

Each research group is willing to share data with other participants, but want to diverge occasionally when they have disagreements about different goals, schemas, and data. Each participant has acceptance rules about what data the site trusts based on its provenance information and also provide reconciliation facilities in order to resolve conflicts[18]. Provenance information associated with updates also allow the system to prioritize updates[17].

## 6 Open Research Problems

Data Provenance have been identified as a major problem in the management of scientific data, yet it is currently not very well-understood. Research shows that provenance is still an exploratory field, several open research problems are exposed, and novel ways are needed to make good provenance recording a reality and to leverage it to its full potential[15].

- Several different systems have been implemented to track the provenance of data sets but little progress has been made to store and retrieve provenance information efficiently. All existing solutions use recursive queries and even recursive tables to represent the work-flows, which do not scale for very large workflow processes and are rather inefficient.

- Scientific systems deal with large volumes of data which is used by the scientists after large number of processing steps. Provenance tracking for such data is challenging because of its volume and because the preferred algorithms used for processing the data also tend to change over time, both as a result of improvements to the software and as a result of changes to the hardware, operating system, and library environment in which the analyses run.

- It is important to keep the state of the database while recording the provenance of a data element because of changes in the databases.

- Provenance information should also be recorded for the queries that involve negation and arbitrary functions.

- Queries that produce same outputs may not convey same provenance information. It should be decided whether we need a more extended, sophisticated query language or annotations should be associated with existing query language.
- Existing security models have potential shortcomings, so a new provenance-aware security model is needed to solve the security problems.
- In curated databases, user interfaces are used to copy data elements from some web page and pasting it into some other web or forms interface to a database. Usually, provenance information is lost during this process. So, these user interfaces should be improved to be provenance aware.
- There is a significant organizational challenge because many organizations do not place a high priority on retaining provenance information since it is expensive but may not provide short-term benefits. Therefore, it is crucial that provenance techniques be inexpensive and provide a clear benefit.
- Provenance data for databases should be according to the schema rather than external annotations.
- Efficient and effective techniques are required to manage large volumes of heterogeneous, distributed workflow and provenance information.
- The problem of querying provenance and extracting useful information from provenance data has been largely un-explored. Also, scientists can obtain a better understanding of their results with the help of provenance data analysis and then creating its insightful visualizations.
- More principled approaches are required to integrate provenance across their independently developed workflow systems.

## References

[1] Bertram Ludascher, Ilkay Altintas, Chad Berkley, Dan Higgins, Efrat Jaeger, Matthew Jones, Edward A. Lee, Jing Tao, and Yang Zhao. Scientific workflow management and the kepler system: Research articles. *Concurr. Comput. : Pract. Exper.*, 18(10):1039–1065, 2006.

[2] Nandita Mandal, Ewa Deelman, Gaurang Mehta, Mei-Hui Su, and Karan Vahi. Integrating existing scientific workflow systems: the kepler/pegasus example. In *WORKS '07: Proceedings of the 2nd workshop on Workflows in support of large-scale science*, pages 21–28, New York, NY, USA, 2007. ACM.

[3] Deana D. Pennington. Supporting large-scale science with workflows. In *WORKS '07: Proceedings of the 2nd workshop on Workflows in support of large-scale science*, pages 45–52, New York, NY, USA, 2007. ACM.

[4] Thomas Heinis and Gustavo Alonso. Efficient lineage tracking for scientific workflows. In *SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1007–1018, New York, NY, USA, 2008. ACM.

[5] Peter Buneman, Adriane Chapman, and James Cheney. Provenance management in curated databases. In *SIGMOD '06: Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 539–550, New York, NY, USA, 2006. ACM.

[6] Peter Buneman and Wang-Chiew Tan. Provenance in databases. In *SIGMOD '07: Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 1171–1173, New York, NY, USA, 2007. ACM.

[7] Peter Buneman, James Cheney, and Stijn Vansummeren. On the expressiveness of implicit provenance in query and update languages. *ACM Trans. Database Syst.*, 33(4):1–47, 2008.

[8] Yogesh L. Simmhan, Beth Plale, and Dennis Gannon. A survey of data provenance in e-science. *SIGMOD Rec.*, 34(3):31–36, 2005.

[9] Luc Moreau, Paul Groth, Simon Miles, Javier Vazquez-Salceda, John Ibbotson, Sheng Jiang, Steve Munroe, Omer Rana, Andreas Schreiber, Victor Tan, and Laszlo Varga. The provenance of electronic data. *Commun. ACM*, 51(4):52–58, 2008.

[10] Laura Chiticariu, Wang-Chiew Tan, and Gaurav Vijayvargiya. Dbnotes: a post-it system for relational databases based on provenance. In *SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 942–944, New York, NY, USA, 2005. ACM.

[11] James Cheney, Peter Buneman, and Bertram Ludäscher. Report on the principles of provenance workshop. *SIGMOD Rec.*, 37(1):62–65, 2008.

[12] Peter Buneman, Sanjeev Khanna, and Wang chiew Tan. Why and where: A characterization of data provenance. In *In ICDT*, pages 316–330. Springer, 2001.

[13] Jihie Kim, Ewa Deelman, Yolanda Gil, Gaurang Mehta, and Varun Ratnakar. Provenance trails in the wings-pegasus system. *Concurr. Comput. : Pract. Exper.*, 20(5):587–597, 2008.

[14] Simon Miles, Paul Groth, Ewa Deelman, Karan Vahi, Gaurang Mehta, and Luc Moreau. Provenance: The bridge between experiments and data. *Computing in Science and Engineering*, 10(3):38–46, 2008.

[15] Susan B. Davidson and Juliana Freire. Provenance and scientific workflows: challenges and opportunities. In *SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1345–1350, New York, NY, USA, 2008. ACM.

[16] Ewa Deelman, Gurmeet Singh, Mei-Hui Su, James Blythe, Yolanda Gil, Carl Kesselman, Gaurang Mehta, Karan Vahi, G. Bruce Berriman, John Good, Anastasia Laity, Joseph C. Jacob, and Daniel S. Katz. Pegasus: A framework for mapping complex scientific workflows onto distributed systems. *Sci. Program.*, 13(3):219–237, 2005.

[17] Todd J. Green, Grigoris Karvounarakis, Zachary G. Ives, and Val Tannen. Update exchange with mappings and provenance. In *VLDB '07: Proceedings of the 33rd international conference on Very large data bases*, pages 675–686. VLDB Endowment, 2007.

[18] Nicholas E. Taylor and Zachary G. Ives. Reconciling while tolerating disagreement in collaborative data sharing. In *SIGMOD '06: Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 13–24, New York, NY, USA, 2006. ACM.

[19] Zachary Ives, Nitin Khandelwal, Aneesh Kapur, and Murat Cakir. Orchestra: Rapid, collaborative sharing of dynamic data. In *In CIDR*, 2005.

[20] Jim Gray, David T. Liu, Maria Nieto-Santisteban, Alex Szalay, David J. DeWitt, and Gerd Heber. Scientific data management in the coming decade. *SIGMOD Rec.*, 34(4):34–41, 2005.

[21] Ilkay Altintas, Oscar Barney, and Efrat Jaeger-Frank. Provenance collection support in the kepler scientific workflow system. pages 118–132. 2006.

[22] Ian Foster, Jens Vckler, Michael Wilde, and Yong Zhao. Chimera: A virtual data system for representing, querying, and automating data derivation. In *In Proceedings of the 14th Conference on Scientific and Statistical Database Management*, pages 37–46, 2002.

[23] Jun Zhao, Carole A. Goble, Robert Stevens, and Sean Bechhofer. Semantically linking and browsing provenance logs for e-science. In *ICSNW*, volume 3226 of *Lecture Notes in Computer Science*, pages 158–176. Springer, 2004.

[24] Carmen Pancerella, John Hewson, Wendy Koegler, David Leahy, Michael Lee, Larry Rahn, Christine Yang, James D. Myers, Brett Didier, Renata McCoy, Karen Schuchardt, Eric Stephan, Theresa Windus, Kaizar Amin, Sandra Bittner, Carina Lansing, Michael Minkoff, Sandeep Nijsure, Gregor von Laszewski, Reinhardt Pinzon, Branko Ruscic, Al Wagner, Baoshan Wang, William Pitz, Yen-Ling Ho, David Montoya, Lili Xu, Thomas C. Allison, William H. Green, Jr., and Michael Frenklach. Metadata in the collaboratory for multi-scale chemical science. In *DCMI '03: Proceedings of the 2003 international conference on Dublin Core and metadata applications*, pages 1–9. Dublin Core Metadata Initiative, 2003.

[25] James D. Myers, Carmen Pancerella, Carina Lansing, Karen L. Schuchardt, and Brett Didier. Multi-scale science: supporting emerging practice with semantically derived provenance. In *In ISWC 2003 Workshop: Semantic Web Technologies for Searching and Retrieving Scientific Data*, 2003.

[26] James Frew and Rajendra Bose. Earth system science workbench: A data management infrastructure for earth science products. *Scientific and Statistical Database Management, International Conference on*, 0:0180, 2001.

[27] Y. Cui and J. Widom. Lineage tracing for general data warehouse transformations. *The VLDB Journal*, 12(1):41–58, 2003.