Data Provenance Survey Based On It's Applications

Ahmed A. Hussein **Esraa H. Alwan** **Majid J. Jawad**
College of Science for Women, Babylon University, Iraq
ahmed_A._Hussein@uobabylon.edu.iq isr.phd@gmail.com,
wsci.majid.jabbar@uobabylon.edu.iq

Abstract

Data provenance is defined as the origin and process history of a derived data item, which is becoming increasingly important for scientific research work. Although provenance can be very valuable for applications, research on data provenance is on its beginning and there is still a series of work that needs to be done. In this paper, some categories, which are provenance granularity, data granularity, data status, provenance computing, semantics of provenance, provenance storage and applications, are proposed to classify and analyze present provenance techniques. In this work we introduce a provenance classification for different applications based on proposed categories, to help the researchers in this field to understand the existing problems and the current challenges and how to enable the current techniques for solving these challenges.
Keywords: Data Provenance, Data Lineage, Data Provenance Taxonomy

الخلاصه

يمكن تعريف تتبع البيانات بأنه عبارة عن الاساس التاريخي والاصلي لعناصر البيانات المستخرجة، سيما وان تتبع البيانات اصبحت وبصورة متزايدة من العمليات المهمة بالنسبة للباحثين. بالرغم من ان عملية التتبع للبيانات مهمة جدا وذات قيمة عالية، لكن البحوث في هذا المجال ما زالت في مراحلها الابتدائية وتعتبر مشكلة جدية و تحتاج البحث عنها بصورة اكثر جدية. هنالك القليل من التصنيفات التي تم اقتراحها لغرض تصنيف وتحليل التقنيات الحالية لمشكلة تعقب البيانات، ومنها تفاصيل المصدر، تفاصيل البيانات، حالة البيانات، حوسبة البيانات، دلالات المصدر، تطبيقات وخزين البيانات. في بحثنا هذا سوف نعمل على تصنيف البحوث المقترحة الحالية اعتمادا على مختلف التطبيقات لمساعدة الباحثين في هذا المجال لفهم المشكلة الحالية وصعوباتها وكيفية تلك التقنيات تمكننا من حل تلك الصعوبات.
الكلمات المفتاحية: تعقب البيانات، بيانات المصدر، تصنيف تعقب البيانات

1. Introduction

Modern science is becoming increasingly dependent on databases which pose new challenges for database technology. With the preservation of the data items in databases, the issues that how and from where information was obtained are particularly important as database technology is employed not just to provide access to source data, but also to the derived knowledge of scientists who have produced and interpreted the data (Buneman *et al.,* 2006). The word "provenance" means origin or source, and is often associated with a piece of art or literature. Merriam-Webste r(http://www.merriam-webster.com) additionally defines provenance as the history of ownership of a valued object or work of art or literature. Data lineage and data provenance have been identified as a major problem in the management of scientific data, since a data set is often useless from a scientific point of view when the derived data sets were produced by scientists using computational means without lineage information. Provenance is useful for scientists in a variety of fields. For example, molecular biology database, which is possibly one of the most sophisticated consumers of modern database technology, often contains data from other databases. Scientists can verify the copied data by tracking provenance.

Although provenance can be very valuable for applications, research on data provenance is on its beginning and there is still a series of work that needs to be done. Present research work solved provenance related problems from different aspects. Storage cost of provenance is much larger than the derived data itself. In order to reduce the storage cost of provenance, Buneman *et al*, (Buneman *et al.,* 2006) study the problem of tracking provenance of scientific data in curated databases that constructed by the scientists who manually assimilate information from several sources and proposed to store a type of provenance called hierarchical transactional provenance, which can reduce the storage overhead by a factor of **5**, relative to a more naive approach.

Cui (Cui and Widom,2003) define the data provenance problem to tracing warehouse data items back to the source data items from which they were derived. Their work study the data warehousing systems by looking not only at the data items in the warehouse, but also to investigate how certain warehouse data items were derived from the sources. Agrawal (Agrawal *et al.,* 2006) built a new kind of database management system called *Trio* project, in which data, uncertainty of the data, and data lineage are considered in an extended relational model and SQL-based query language. Ram (Ram et al., 2006) developed an ontological model of provenance called the W7 model that conceptualizes data provenance as a combination of seven interconnected elements including "what", "where", "when", "how", "who", "which", and "why", which captures and represents the semantics of data provenance and lineage. In their work, provenance is described as the history of the derived data, which includes its origin, key events that occur over the course of the lifecycle, and other related information associated with the creation, processing, and archiving of derived data items. Reference (Karvounarakis, *et al,* 2010) studied how to query provenance in an application-independent way and a general provenance querying language ProQL is proposed. Network provenance is defined as the ability to issue queries over network meta-data, which is important for network accountability, forensic analysis, and failure diagnosis Zhou (Zhou, *et al.,* 2010) presented the design and implementation of ExSPAN, a generic and extensible framework that achieves efficient network provenance in a distributed environment. Panda (Ikeda and Widom, 2009) developed a general-purpose open-source system that can be configurable to be used for a variety of applications. Panda project can be formalized as a model, which is represented as a graph structure connecting input and output data elements.

The rest of this paper is organized as follows. In section 2, we present the categories based on generic view of provenance. Then section 3 applies proposed categories to formalize and analyze a set of selected data provenance techniques for different applications.

2. Categorization of provenance techniques

In this section, a general categorization of provenance management systems is presented. A current researches can be classified into the seven categorizes, which are ***provenance granularity***, ***data granularity***, ***data status***, ***provenance computing***, ***semantics of provenance***, ***provenance storage*** and ***applications***. A summary of the category is given in Figure 1.

2.1 Provenance granularity

Granularity is a term used extensively in the software programming arena. It is also applicable to the storage of provenance. The granularity of provenance can be divided to ***coarse-grained*** and ***fine-grained***.

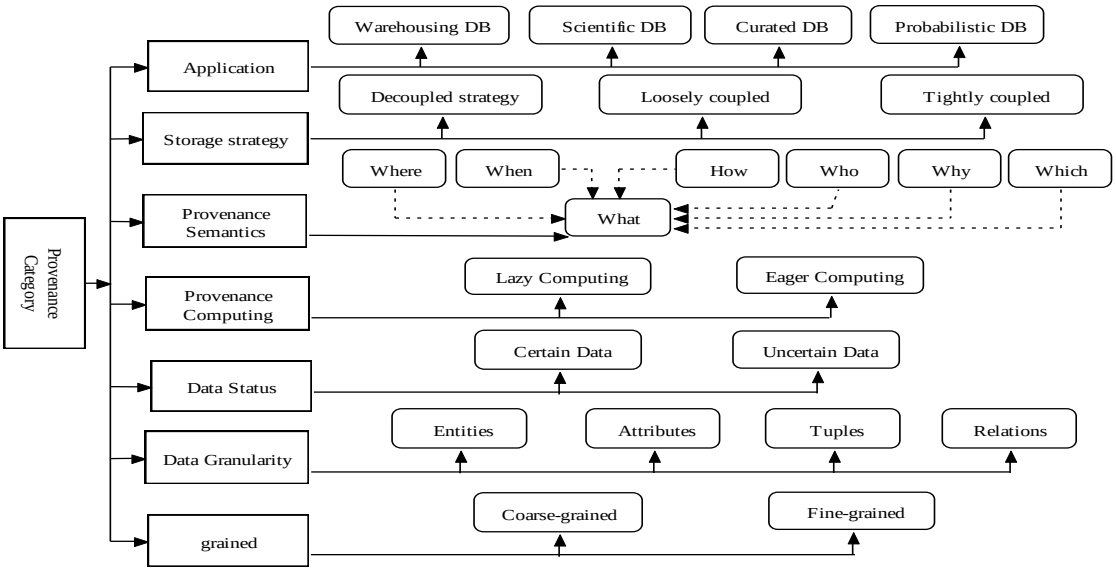

Figure 1. The Provenance Category.

2.1.1 Coarse-grained

Coarse-grained data structures present in the forms of workflows, which combine pieces of information into one piece. In general, Coarse-grained provenance keeps each transformation for deriving the data. Although it is useful to Figure 2 out the overview of the processing, it is not enough for tracking the origins of data.

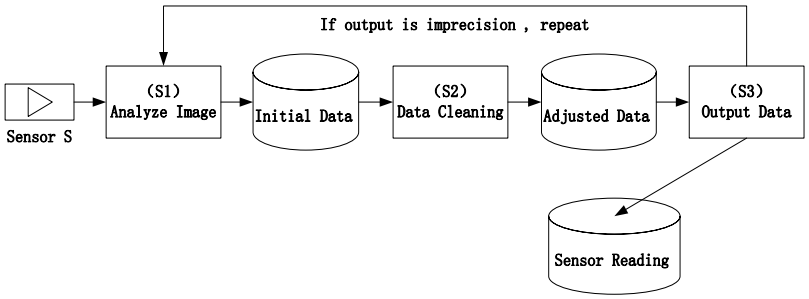

Figure 2. An Example of Coarse-Grained Provenance.

A simple example of workflow is depicted in Figure.2. Arrows denote the flow of data, while boxes are used to indicate data processing steps. This workflow describes the steps taken to get the correct readings of sensors. The workflow starts with step (S1) which analyzes the image from the sensors. The second step (S2) is cleaning the initial data gotten from step (S1) and obtain the adjusted data. Step (S3) involves outputting the adjusted data, so the readings of the sensors are received. In general, external processes do not possess good properties for a detailed analysis of the transformation since such details are typically hidden. Hence, the workflow provenance for this step is usually coarse-grained. Only the input, output and the transformations used by an external process are recorded (Tan, 2007).

2.1.2 Fine-grained

In contrast, fine-grained provenance records the origins for each result data. Obviously, fine-grained provenance shows more detailed information about how the result data are derived. However, a problem of fine-grained provenance is storage. The storage of fine-grained provenance increases based on the size of result data while that of coarse-grained provenance does not(Tan, 2007).

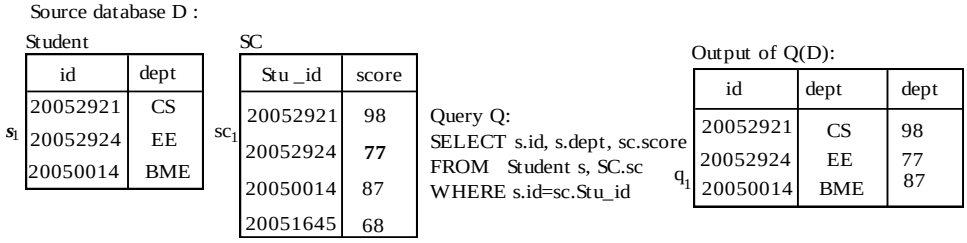

Figure 3. An Example of Fine-Grained Provenance.

Figure 3 shows an example of fine-grained provenance. Suppose *D* and *Q* are the database and query, respectively. The result of executing *Q* against *D* is also shown on the right of the same figure. The source tuples s_1(*20052924, EE*) and sc_1(*20052924, 77*) contribute to the output tuple q_1(*20052924, EE, 77*) according to *Q*. In particular, observe that some source tuples, such as (*20050014, BME*), play no role in contributing to the output tuple (*20052924, EE, 77*) according to *Q*. Thus the data provenance (fine-grained provenance) of tuple q_1 is tuples s_1 and sc_1.

2.2 Data granularity

A database is a set of data structures for organizing and storing data. In any data model, e.g. DBMS, we have a set of principles for exploiting such data structures for information system applications within organizations. Dr. Codd (Codd, 1970) originally borrowed from the terminology of mathematics to denote elements of the data model. Columns of tables are known as attributes. Rows of tables are known as tuples (Celko, 1999; Davies,2004).This section provides with the brief concepts of data granularity.

An entity is defined as a thing that an organization recognizes as being capable of an independent existence and can be uniquely identified. An entity is an abstraction from some domain. When we speak of an entity, we normally speak of some aspects of the real world which can be distinguished from other aspects.

Each real-world individual of a class is represented by a row of information in a database table. The row is defined in the relational model as a tuple that is constructed over a given scheme. Mathematically, the tuple is a function that assigns a constant value from the attribute domain to each attribute of the scheme. Notice that because the scheme is a set of attributes, we could show them in any order without changing the meaning of the data in the tuple. For example in Figure 3, the tuple s_1 in relation *Student* is (*20052924*, *EE*).

Data must be stored in some fashion in a file for it to be useful. In database there have been three basic camps of "logical" database models — hierarchical, network, and relational — three ways of logically perceiving the arrangement of data in the file structure. A relational model is a way of tying objects together to get new information that exists apart from the particular objects. Each relation must have a primary key, which is to enforce the property that duplicate rows are forbidden in a relation. A

primary key is one or more columns of a table whose values are used to uniquely identify each of the rows in a table. Also in Figure 3, attribute *id* is the primary key for relation *Student*.

2.3 Data status: Certain data and uncertain data

Traditionally, databases have required data to be modeled in terms of precise values. However there are many applications where uncertainty, or imprecision in values is inherent or desirable (Cheng, 2006; Singh, 2009).

The growing importance of several new application areas—information extraction on the web, information integration, scientific databases, sensor and RFID data management—create an increasing need to deal with uncertain data. Existing database management systems (DBMSs) do not support uncertain data, hence applications are left to either: (1) Clean away the uncertainty so the data can be stored in and queried by a traditional DBMS; or (2) Handle the uncertainty at the application layer, and use a DBMS only for traditional data management. While the former approach results in loss of information and errors compounding when operations are performed, the latter imposes a significant burden on applications.

In light of applications that use uncertain data, the DBMS should be able to store, manage and query uncertain data. Obviously, the uncertainty associated with the data and corresponding provenance should be considered.

The operations of selection, projection and joins are widely used for querying traditional databases. Unfortunately, the semantics of these operations are not clear for uncertain data. For example, consider a selection query $x>6$ over a precise database table *T*. This query will return all the tuples in which the attribute *x* is greater than 6. If the attribute *x* is uncertain, this selection condition can be partially true for many tuples. For each tuple, instead of a precise true or false answer, each tuple may be the true answer with a possibility. There are three main models of uncertain data in databases: attribute uncertainty, correlated uncertainty and tuple uncertainty. In attribute uncertainty, each uncertain attribute in a tuple is subject to its own independent probability distribution. In correlated uncertainty, multiple attributes may be described by a joint probability distribution. In tuple uncertainty, all the attributes of a tuple are subject to a joint probability distribution.

The relationship between uncertainty and lineage is that lineage can be used for understanding and resolving uncertainty. To draw a loose analogy with web search, correctness of answers returned by a search engine are uncertain, reflected by their ranking. Search engines typically provide lineage information including at least a URL and text snippet, and users tend to consider both ranking and lineage to determine which links to follow. More generally, any application that integrates information from multiple sources may be uncertain about which data is correct, and the original source and derivation of data may offer helpful additional information.

2.4 Computing data provenance: lazy and eager approach

Currently, different data provenance computing approaches are devised in variable settings in order to trace data's provenance. The existing techniques for computing data provenance can mainly be divided into two categories: lazy and eager approache (Tan, 2004; Cency, 2009; Cui and Windom, 2000; Larissa *et al.*, 2014; Samir *et al.*, 2014).

The lazy approach is the inverse approach for computing provenance. When using an inverse approach, a query is generated and executed to compute the provenance only when needed, thus the lineage is not explicitly computed and stored which results in less provenance storage space. The lazy approaches can be used in the queries and user-defined functions in databases that can be inverted automatically or by explicit functions. The main method of the eager provenance tracing approach is to propagate annotations along data transformations. Many current provenance systems that use annotations have adopted XML for representing the lineage information. Thus the eager approach has been given several names, such as metadata support, source tagging, the attribution approach, or annotations. An advantage of using an eager method to compute data provenance is that the source databases need not be probed since the provenance can be fully determined by the annotations associated with a piece of output data. A disadvantage is that the eager approach incurs additional overhead to compute the output and store annotations in the output (Tan, 2004).

2.5 Semantics of data provenance

This section investigates the semantics of data provenance based on reference (Ram et al., 2006) ,which represents provenance as a combination of seven elements including "What", "Where", "When", "How", "Who", "Which" and "Why". Figure 4 provides an overview of the W7 provenance model.

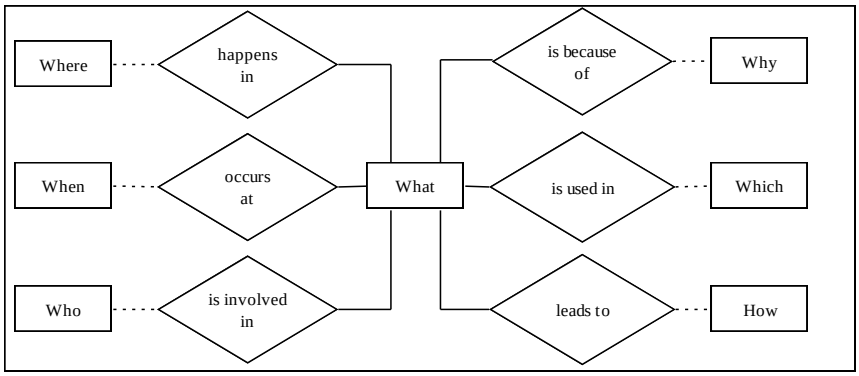

Figure 4. Overview of W7 Provenance Model.

"*What*" is the anchor of the W7 provenance model. "*What*" is a sequence of events that affect a data object during its life time. The other provenance components are semantically related to "*What*" and describe various details about the events.

"*Where*" denotes a set of locations where various events happen. According to the W7 provenance model, the most common forms of representing locations are physical, geographical and transaction locations. Physical locations specify the position of places or points based on a global coordinate system. Geographical locations signify an area or boundaries governed by a common law and are normally organized hierarchically. Transaction location links a data object to its location in a server or database. This concept of transaction location is important since data may travel between information sources due to events such as storage and transfer. The transaction location can often be represented by a URI, and it can be typed into source and destination.

"*When*" represents a set of timestamps associated with various provenance events. The W7 provenance model records the occurrence time of various events that affect data object during its lifetime. The W7 provenance model also records the durations of the events, which includes start point and end point. Associating a timestamp with each event provides a detailed timeline of the events and enables the ability of reconstructing the history of the data object.

"*How*" records the actions which lead to the events' occurrences. Actions are performed by agents to obtain a desired outcome, thus actions are causes of event and events are occurred by the actions. Normally information about actions includes preconditions, methods, inputs, outputs and sources. Actions are classified into primitive actions and complex actions. Primitive actions are defined as the ones that can not be decomposed. Complex actions are composed of primitive actions.

"*Who*" refers to agents who bring about the events. Here agents refer to persons, organizations, software applications, etc. Each agent plays a certain role involved in the action that lead to the occurrence of the events. For example, a government agent may play the role of supervisor in creating a script.

"*Why*" captures a set of reasons for various provenance events. The W7 provenance model specifies *beliefs* and *goals* as two subsets of "*Why*". Beliefs represent the knowledge of the world, which are classified into assumptions and hypotheses. Goals are the intentions that an agent wants to achieve.

"*Which*" describes which devices are used in the derivation of a data object, where derivation includes creation, analysis, and transformation. Devices can be distinguished into *instruments* (e.g. equipments and hardware) and *applications*. The information about a device is classified into *description*, *function* and *settings*. When an event involves a device, "*Which*" captures the details about the device.

2.6 Provenance storage

Provenance can grow to be much larger than the data itself. The manner in which the provenance metadata is stored is important. In this paper, we classified three main principal storage strategies: the decoupled, the tightly coupled and the loosely coupled storage strategy. Coupling or dependency describes the degree of how data provenance relies on other data. Coupling can be low when data provenance is stored independently. Coupling can also be high when data provenance is complexly dependent on other data.

When adopting decoupled strategy, provenance information is stored in one or many provenance repositories. These repositories are dedicated to store only provenance data, and each repository usually has very limited knowledge of any other repository. A decoupled strategy usually allows changes to be made to any repository without having an effect on any other repository.

In loosely coupled strategy, there is a group of mapping schemas which describe how to trace provenance using present stored data. And the data are stored and operated independently of each other.

Tightly coupled strategy recodes and stores provenance directly associated with the data which are dependent on each other. Tightly coupled strategy usually gets an efficient performance. Since provenance storage strategy is designed for specific applications using corresponding background knowledge, thus the transplant ability of tightly coupled strategy is not good.

2.7 Database applications

We have classified the database applications which show that storing data provenance is essential. In different database applications, there are always different methods for encoding and storing provenance based on different settings. The meaning of data provenance is also different for specific scenarios.

2.7.1 Data warehousing

Data warehouse is a repository of an organization's electronically stored data[19]. A data warehouse houses a standardized, consistent, clean and integrated form of data sourced from various information systems used by the organization, structured in a way to specifically address the reporting and analytic requirements. Figure 5 shows a simple architecture for a data warehouse. End users directly access data derived from several source systems through the data warehouse. Enabling provenance tracing in a data warehousing environment has several benefits, including in-depth data analysis and data mining, authorization management, view update and efficient data warehouse recovery.

2.7.2 Scientific database

Scientific database can be represented in many fields of sciences such as Astronomy, Biochemistry, Biology, Chemistry, Computer Science, Engineering, Environment, Geology, Marine Science, Mathematics, Medical Sciences, Physics and Statistics. In recent years, the nature and use of scientific database, the conditions under which scientific data are produced, distributed, and managed, and the role of scientists and other actors in these processes have been changing rapidly. These changes are partly a result of the revolution in computational capacity and connectivity and advances in hardware and software that have provided scientists with a greatly increased capacity for data gathering, analysis, and dissemination. They are also related to the emergence of new questions in scientific research that require different types of data. Scientists routinely rely on information from a wide array of sources to help them plan their research work. Such information needs might include reaction pathways, information properties and sourcing. Scientific data may be processed into multiple derivative work, each having a different intended use or audience**.** Data provenance has been identified as a major problem in the management of scientific data. Without provenance information, a data set is often useless from a scientific point of view.

2.7.3 Curated database

Curated databases are referred as databases that are manually created, in which annotations, corrections and transfer of data from other sources by scientists through studying and analyzing information from many sources. Since the convenience of publishing data on the web, the number of new curated databases for scientific research has increased greatly. In reference (Buneman et al., 2008), some notable examples of curated databases include: 1) UniProt (formerly called SwissProt), which forms the standard reference for protein sequence data in molecular biology. 2) The CIA World Factbook, which is probably the most widely used source of demographic data. 3) The IUPHAR receptor database, which describes the molecules that transmit information across the cell membranes. This database is typical of a very large number of small curated biological databases. Curated databases present challenges for database research, since curated databases are heavily cross-referenced with each other, and include data from other databases (Buneman *et al.*, 2006; Buneman *et al.*, 2008).

Provenance information concerning the creation, attribution, or version history of such data is crucial for assessing curated databases' integrity and scientific value.

2.7.4 Probabilistic database

A probabilistic database is an uncertain database in which the possible worlds have associated probabilities. Probabilistic data management is currently an active area of research, which can distinguish between the logical data model and the physical representation of the data (Antova *et al.*, 2007; Davli and Suciu, 2007). Probabilistic databases can store types of information that cannot be represented using the relational model. It can also be viewed as generalizations of relational databases. Any relational database can be represented without loss of information by a probabilistic database (Cavallo and Pittarelli, 1987). In this paper, we discuss the Trio project in which uncertainty and lineage is combined together.

3. Some Data Provenance Techniques

This section identifies some data provenance techniques. Some researchers are selected to provide a comprehensive overview of some presented techniques on data provenance.

3.1 Lineage tracing for general data warehouse transformations (LTDW)

Cui (Cui and Widom ,2003) trace lineage information for view data and general transformations in data warehouses. Data warehousing systems integrate information from operational data sources into a central repository to enable analysis and mining of the integrated information (Chaudhuri and Dayal, 1997; Loment and Widom, 1997).

LTDW system presents a set of techniques for data warehouse provenance tracing when the warehouse data is generated through a graph of general transformations. LTDW system takes advantage of known structure or properties of transformations that hold frequently in practice which can be specified easily by transformation authors. The tracing algorithms apply to single transformation, to linear sequences of transformations, and to arbitrary acyclic transformation graphs by recording the relational queries used to construct materialized views in a data warehouse in order to find the source data that contributed to the given data item.

LTDW system considers three kinds of properties and provides corresponding algorithms that trace data provenance using these properties. The first provenance tracing method is based on how a transformation maps input data items to output. The second property of transformation for tracing provenance is schema mappings since there may be one or more schema mappings for a transformation which specify how output attributes relate to input attributes. The third provenance tracing property of a transformation is that a transformation may be accompanied by a tracing procedure or inverse transformation, which is the best case for provenance tracing. When a transformation exhibits many properties, the researchers determine the best one for lineage tracing based on a property hierarchy.

Cui (Cui and Widom,2003) considered fine-grained (or instance-level) lineage to retrieve the actual set of source data items that derived a given warehouse data item. Coarse-grained provenance information (e.g. schema mappings) can be used in support of fine-grained provenance tracing techniques. Provenance tracing is modeled as inverse queries for transformation sequences which is defined as $T_1 \circ \cdots \circ T_n$, where each I_k is the intermediate result output from T_{k-1} and input to T_k, as shown in Figure 6. It is clear

that LTDW system present a tightly coupled provenance storage strategy since each transformation stores provenance information directly which is dependent upon each other. Storing all intermediate results $I_2, \ldots, I_n$ for provenance tracing is inefficient since the large number of tracing procedure calls when inverse query each transformation in the sequence lead to low performance efficiency and high storage cost. This problem is relieved by combining beneficial adjacent transformations repeatedly in a sequence through a greedy algorithm until no more beneficial combinations are found.

The semantics of provenance represented in LTDW system include "*What*", "*Where*" and "*How*" provenance. LTDW system can get provenance tuples for a derived data item based on the transformation graph through inverse query, and the resulted tuples are the "*What*" and "*Where*" provenance for the derived data item. The transformation graph describes how the data is generated which is the "*How*" provenance.

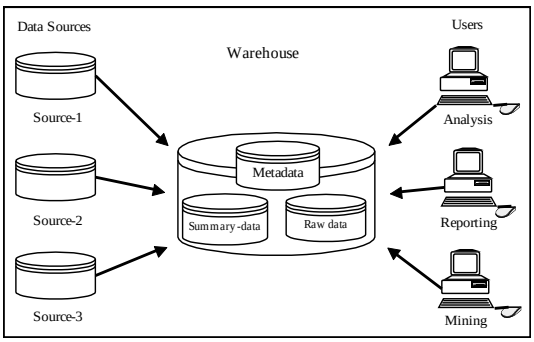

Figure 5. A simplified architecture for data warehouse.

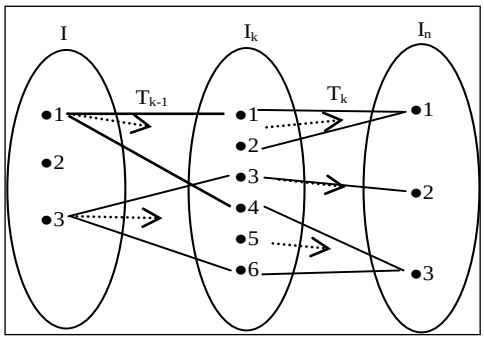

Figure 6. Lineage tracing for transformation sequences.

3.2 Databases with uncertainty and lineage (ULDBs)

Agrawal *et al.* are building a new kind of database management system called *Trio* project, in which data, uncertainty of the data, and data lineage are considered in an extended relational model and SQL-based query language. Trio project is motivated by the need for coexisting of these three aspects in one system, and detailed numerous potential applications including scientific data management, data cleaning and integration, information extraction systems, in which there exists uncertainty in collected data.

Benefit from the initial vision of the Trio project, Widom(Widom, 2005) defined a probabilistic database model called ULDBs(Databases with Uncertainty and Lineage). ULDBs extend the relational model with simple forms of uncertainty combined with lineage and yield nice properties and strong expressiveness. Uncertainty is expressed by tuples that include several alternative possible values for some (or all) of their attributes with optional confidence values associated with each alternative. Based on alternative value and confidences, each ULDB represents multiple possible-instances (i.e. possible-worlds), where a possible-instance is a regular relational database. Lineage associated with a data item describes the data item's derivation, thus provenance in Trio system is described in fine-grained since each tuple has its provenance which can pin down the source data contributed to the generation of each resulted data. Trio system also adopts an eager strategy in computing data provenance, since when a relational query is performed, each tuple in the derived result is annotated with its lineage formula.

Lineage formula associated with a derived data item shows the uncertain data items contributed to the derivation which is the "*What*" provenance. Lineage formula

also refers to internal data provenance within the ULDB, or external data provenance which is outside the ULDB, or to other data provenance derived by programs or devices, thus lineage formulas in ULDBs describe the "*How*" and "*Where*" provenance.

The provenance of the tuples in ULDBs is stored in an auxiliary table and derived tuple and corresponding provenance are associated with unique tuple-ID, which shows that ULDBs adopts a loosely coupled provenance storage strategy. ULDBs records one level provenance, meaning only the tuples from which a tuple was directly derived are stored in the provenance relation. A database-wide unique tuple ID is used to identify tuples (Glavic and Dittrich, 2007).

A SQL-based query language for ULDBs is proposed which is named TriQL. TriQL modifies the semantics of SQL to take uncertainty and lineage into account, and introduces new constructs to query uncertainty and lineage directly (Benjelloun and Das, 2006). Intuitively, the result of a relational query *Q* on a ULDB *U* is a result *R* whose possible-instances correspond to applying *Q* to each possible-instance of *U*. Confidence values of a query result can be computed with the derived result's lineage formula. TriQL query results can either be *stored* or *transient*. Stored results are placed in a new persistent table, and corresponding lineage information is also stored persistently. And neither the result data nor corresponding lineage are persistent for transient query (Widom, 2008). .

3.3 Provenance management in curated database (PMCD)

Curated databases are constructed by the scientists who manually assimilate information from several sources. Reference Reference (Buneman et al., 2006) studied the problem of tracing and managing provenance which describes the user actions involved in constructing a curated database.

Buneman (Buneman *et al.*, 2006) that there are four main operations in curated databases, which are insert, delete, copy, and paste actions, as shown in Figure 7. A curated database is constructed through these actions by the users. In order to maintain the provenance of a curated database, a provenance-aware application for browsing and editing databases is proposed to capture a sequence of actions which construct the curated database. When editing data in local database *T*, provenance links are stored in an auxiliary provenance database *P*. Data locations in *T* are related with locations in previous versions of *T* or in external source databases *S* with stored provenance in *P*, thus PMCD adopts a loosely provenance storage strategy.

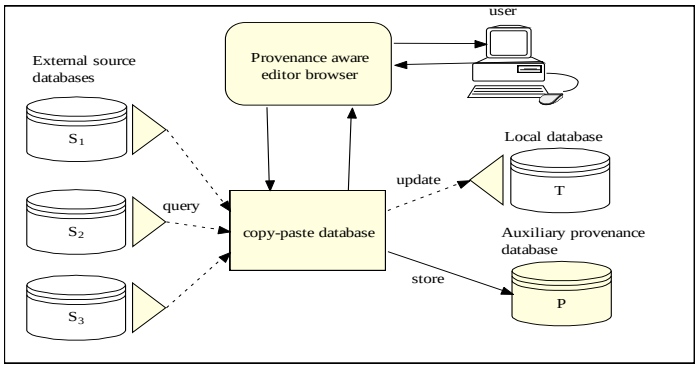

Figure 7. Provenance system Design in PMCD.

When recording the detail provenance of curated databases, if there are many edit operations for updating the curated databases between different versions, the rising problem is the increasing provenance storage cost with the large number of edit operations. The main technical challenge of PMCD is to minimize the overhead required to store the lineage information. Lineage query performance isn't considered which is assumed to be relatively rare. In order to reduce the amount of storage space needed for lineage information, reference (Buneman et al., 2006) proposed optimizations for two kinds of provenance structure, which are transactional provenance and hierarchical provenance. When editing curated databases, multiple updating operations can be grouped into transactions. Thus lineage storage cost can be reduced through storing transactional provenance instead of operation provenance. Hierarchical relationships between data items can be used to optimize the provenance storage, which is the basic idea of hierarchical provenance. Provenance storage cost is compressed with hierarchical provenance through summarizing transformations. For example, if all tuples in a relation are copied, hierarchical provenance will store the transaction of copying the relation instead of the transactions of copying the tuples. Combining transactional and hierarchical provenance, the provenance storage overhead is reduced by a factor of 5 compared with no optimization is used.

Since the provenance kept in PMCD includes the process of generating and updating a derived data item, semantics of provenance in PMCD contains the "*What*", "*Where*" and "*How*" provenance. Provenance in PMCD also records the users of the operating actions, which represents the "*Who*" provenance. PMCD keeps the fine-grained provenance for each derived data item through recording the process of generating and updating the data items.

3.4 Understanding the semantic of data provenance to support active conceptual modeling (PROMS)

Reference (Ram *et al.,* 2006) investigated the semantics or meaning of data provenance and developed a generic model named the W7 model that represents data provenance as a combination of seven interconnected elements including "What", "When", "Where", "How", "Who", "Which" and "Why". Each of these elements can be used to track provenance. The W7 provenance model is a generic model of data provenance and is intended to be easily adaptable to represent domains or applications which have specific provenance requirements in conceptual modeling. Different applications decide the collecting semantics of provenance based on their scenarios.

Based upon Mario Bunge's view that a history of a thing is a sequence of events or state changes that happen to it (Bunge,1977), provenance is kept by recording all events that affect data. In database applications, these events center on the lifecycle of data which includes creation, updates, and access of data, as shown in Figure 8.

In W7 provenance model, provenance is defined as a n-tuple *P* = (*What, When, Where, How, Who, Which, Why*), where "*What*" denotes the sequence of events that affect the data object; "*When*" denotes the set of all timestamps related with the events; "*Where*" denotes the set of all locations of the events; "*How*" denotes the set of all actions leading up to the events; "*Who*" denotes the set of all agents involved in the events; "*Which*" denotes the set of all devices; "*Why*" denotes the set of all decisions for the involved events.

Based on the W7 provenance model, a PROvenance Management System (PROMS) is designed and developed for harvesting and using data provenance. The

architecture of PROMS is shown in Figure 9 PROMS records user-provided provenance using templates based on the W7 model. PROMS also supports semi-automatic harvesting of provenance by extracting information stored in electronic notebooks with little manual intervention through natural language processing, which can be seen as lazy provenance computing approach. The harvested provenance is stored in the data provenance knowledge base which is implemented over a relational database. Data and its provenance can be retrieved by users via a web-based graphical user interface. The Provenance Navigation Module allows the user to view and navigate provenance in a convenient way, which enables the user to query data via the provenance (Ram *et al.,* 2006). PROMS focuses on the semantics of the provenance which is defined as a sequence of events affecting the data object and events can be described with different data granularity in different scenarios, thus provenance in PROMS is specified at different data granularity levels. The stored provenance and data are logically independent, thus PROMS adopts a decoupled provenance storage strategy.

PROMS has been used to harvest and store data provenance for WIKI's. The Wikipedia contains more than 1,400,000 articles and most of these articles undergo frequent changes. PROMS monitors the provenance of Wikipedia pages in order to track changes made to a page. PROMS also tracks "*Who*" made the changes, "*How*" and "*Why*" the changes were made and

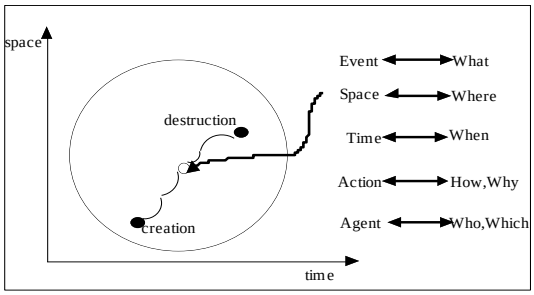

Figure 8. Bunge's View.

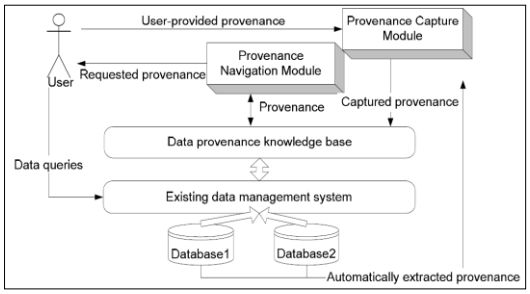

Figure 9. Architecture of PROMS.

at "*When*" time. The recorded provenance is used to generate warnings about potential vandalism threats to Wikipedia pages.

3.5 Querying data provenance (ProQL)

Previous introduced research work mainly investigates the problems of how to model for provenance, compute provenance, etc, and few work focus on the problem of querying provenance. Karvounarakis (Karvounarakis, *et al.*, 2010) studied how to query provenance in an application-independent way and a general provenance querying language ProQL is proposed.

Provenance describes the process of how a derived data item is produced, which is useful for evaluating the existence or scores of derived data items. Thus provenance information is especially important for many advanced data management operations (e.g., incremental maintenance, trust assessment, debugging schema mappings, keyword search over databases, or querying answering in probabilistic databases). Provenance in reference (Karvounarakis, *et al.*, 2010) was described as hypergraph or lineage formula, which are treated as equal. A provenance hypergraph describes the provenance of derived data items through modeling the relationships among tuples in source and derived tables. Lineage formula is the annotation attached to the tuple describing how it

came to be, which is equivalent to the hypergraph representation. Karvounarakis (Karvounarakis *et al.,* 2010) considered the most general formalism for tuple-based provenance, semiring provenance.

A query language for provenance named ProQL is developed, which can express many types of queries. ProQL is developed for the settings of collaborative data sharing systems (CDSSs), which is also useful in supporting a wide variety of applications with derived data items. Four basic ProQL primitives are introduced through a use case, which are "FOR", "WHERE", "INCLUDE PATHS" and "RETURN". ProQL can be applied to query provenance hypergraphs to get sub-hypergraphs which can describe the ways of a tuple derived, what data derives from specific mapping, tuples with the same provenance as specific tuple, etc. Provenance returned by ProQL satisfies the semantics of "What", "Where" and "How" provenance. When a subgraph is returned through ProQL querying the provenance hypergraph, ProQL can also compute semiring annotations (lineage formulas) for sets of tuples. Primitives "EVALUATE *semiring* OF" and "ASSIGNING EACH" are developed to compute semiring annotations. Use cases described for computing semiring annotations include derivability, trust, confidential level, weight, cost, lineage and probability. Initial implementation of ProQL queries over acyclic provenance graphs.

ProQL runs over a SQL DBMS and provenance is stored in a set of relations in a DBMS. Each hyper-edge of hypergraph is stored as a table. ProQL computes provenance paths of provenance subgraphs through joining the tables. In the context of CDSS, ProQL firstly converts the schema mappings into a provenance schema graph and match the ProQL query against the provenance schema graph to identify nodes that match path expressions. In order to improve the efficiency of querying provenance, ProQL uses ASRs to index data provenance, which creates materialized views for joins among provenance relations that correspond to paths of mappings along some derivations.

3.6 Efficient querying and maintenance of network provenance at internet-scale (ExSPAN)

Network provenance is defined as the ability to issue queries over network meta-data, which is important for network accountability, forensic analysis, and failure diagnosis. Zhou (Zhou *et al.,* 2010) presented the design and implementation of ExSPAN, a generic and extensible framework that achieves efficient network provenance in a distributed environment.

ExSPAN tracks network provenance that explains the existence of any network state in the context of declarative networking, in which network protocols can be modeled as continuous queries over distributed streams and specified concisely in a declarative query language. In ExSPAN, declarative networks are specified using Network Datalog (NDlog), which is a distributed recursive query language used for querying network graphs. ExSPAN supports three levels of granularity for network provenance which are tuple-level, node-level, and trust domain level. Tuple-level network provenance records the maximum amount of provenance information, correspondingly results in the largest communication overhead. Node-level network provenance reflects which elements of the network are involved in producing a given tuple. For trust domain level network provenance, nodes within a trusted domain are grouped sharing a domain identifier, which encodes the minimum amount of provenance information but useful for some specific applications, such as access control policies. Network provenance for tuples in ExSPAN is represented as provenance graph

or provenance algebraic formulas. ExSPAN supports centralized or distributed provenance storage. With centralized provenance storage, the entire provenance is attached with the derived tuple, which causes high aggregate bandwidth utilization. When adopting distributed provenance storage fashion, ExSPAN supports value-based distributed provenance and reference-based distributed provenance. For value-based distributed provenance, each derived tuple contains its entire provenance when transmitted between nodes, which is an eagerly provenance propagating method. For reference-based distributed provenance, provenance is dispersed among network nodes through shipping markers (identifiers for tuples) with transmitted tuples, which causes little communication overhead but requires a distributed querying protocol to compute provenance.

Provenance of a tuple in ExSPAN is described as an acyclic graph stored in two relations *prov* and *ruleExec*, which are distributed and partitioned across all nodes in the network. Thus provenance in ExSPAN is tightly coupled stored. Each entry in *prov*, which maintains provenance information, represents a directed derivation of a tuple. Entries in *ruleExec* maintain the meta-data of the executed rule, including the actual executed rule, input tuples, and the location where the rule stored. For reference-based distributed provenance, the provenance of tuple can be discerned through traversing provenance graph with querying *prov* and *ruleExec* recursively, which compute tuple's provenance lazily. Zhou (Zhou *et al.*, 2010) proved that for any NDlog program, additional NDlog provenance maintenance rules can be rewritten automatically as described in the use cases. Provenance in ExSPAN is described as provenance graph, which represents the provenance semantics of "What", "How" and "Where". In order to improve the efficiency of provenance querying, ExSPAN adopts to cache provenance querying results and different traversal order for variable applications.

3.7 System for provenance and data (Panda)

Panda project at Stanford (Ikeda and Widom, 2009) aims to develop a model and system that offers users a full range from fine-grained to coarse-grained provenance by combining these two types of provenance. Panda also provides interfaces that coupled with external data sources, process and systems to support mechanisms for provenance capture, storage, operations and queries.

Panda project develops a general-purpose open-source system that can be configurable to be used for a variety of applications. Panda project can be formalized as a model, which is represented as a graph structure connecting input and output data elements. Panda defines a uniform interface to create and manipulate provenance by manually and automatically methods. Panda uses a set of built-in operations and develops a query language to analyze over provenance information. The provenance querying language in Panda has specific features for additional expressiveness and must be flexible to find efficient query execution plans. The provenance semantics in Panda of built-in operations through backward tracing a given derived data satisfy "Where did the derived data come from?", "What selected data elements contributed to destination?", "How process nodes operate through the backward or forward tracing" and "For what reason to design the model". The choice between eager and lazy computation for provenance in Panda system must be decided based on specific applications. Eager computation may lead to low performance because of overhead of recording the source data items that contributing to destination item. And lazy computation may lead to cost of computations because of re-computing the entire data set. Overall, there is a suite of interesting optimization problems involving decisions

about intermediate data sets and eager versus lazy computation that need to be solved for Panda system.

3.8 Ariadne: Managing Fine-Grained Provenance on Data Streams

Boris (Boris *et al.,* 2013) proposed a novel propagation-based approach for provenance generation, called operator instrumentation. Their approach annotates regular data tuples with their provenance while they are being processed by a network of streaming operators. Propagation of these provenance annotations is realized by replacing the operators of the query network with operators that create and propagate annotations in addition to producing regular data tuples. They referred to this transformation as operator instrumentation. The key idea behind the operator instrumentation approach is to extend each operator implementation so that the operator is able to annotate its output with provenance information based on provenance annotations of its inputs. Under operator instrumentation, provenance annotations are processed in line with the regular data. That is, the structure of the original query network is kept as is (operators are simply replaced with their instrumented counterparts). Thus, most issues caused by non-determinism are dealt with in a rather natural way, since the execution of the original query network is traced. The only drawback of operator instrumentation is the need to extend all operators. However, this extension can be implemented with reasonable effort. With operator instrumentation, provenance can be generated either eagerly during query execution or lazily upon request.

The proposed approach can also be used to compute the provenance of a part of the query network by only instrumenting a subset of the operators. Previous annotation propagation approaches for fine-grained stream provenance are restricted to one-step provenance, i.e., annotating output tuples from an operator with their provenance from the operator's input. Boris G. et al approach is more general and flexible; provenance can also be propagated through several operators or even a complete query network. By lifting this restriction they are able to overcome many of the shortcomings of these approaches including large storage overhead (tracking provenance through a path in the query network requires storage of all streams on the path) and expensive retrieval (queries over provenance require recursive tracing using the single-step provenance). They represent provenance as sets of tuple identifiers during provenance generation. Querying provenance is supported by reconstructing complete input tuples from the identifier sets using a new operator called p-join. This is achieved by temporarily storing input stream tuples for the reconstruction.

A number of optimizations enable them to decouple provenance management from query processing: The Replay-Lazy optimization reduces the run-time overhead of provenance computation by propagating a concise superset of the provenance and lazily replaying a query network to reconstruct its provenance. The Lazy-Retrieval method avoids reconstructing provenance for retrieval if parts of the provenance will not be needed by the query. Furthermore, they devise a number of compression schemes to reduce the computation cost.

3.9 Efficient Stream Provenance via Operator Instrumentation (ESvOI)

Boris (Boris *et al.,* 2014)proposed a novel propagation-based approach for provenance generation, called operator instrumentation. They use a simple definition of fine-grained provenance that is similar to Lineage in relational databases

(Chency, 2009). Their approach annotates regular data tuples with their provenance while they are being processed by a network of streaming operators. Propagation of these provenance annotations is realized by replacing the operators of the query network with operators that create and propagate annotations in addition to producing regular data tuples. Their approach is more general and flexible as it allows a) both eager or lazy provenance generation and b) direct provenance propagation for partial as well as complete query networks. They represented provenance as sets of tuple identifiers during provenance generation. A number of optimizations enable them to decouple provenance management (generation and retrieval) from query processing.

We can summaries their contributions as follow:

—They introduce a novel provenance generation technique for DSMS based on annotating and propagating provenance information through operator instrumentation, which allows generating provenance for networks and subnetworks without the need to materialize data at each operator.

—They propose optimizations that decouple provenance computation from query processing

—They present Ariadne, the first DSMS providing support for fine-grained multi-step provenance.

—They provide an experimental evaluation of the proposed techniques using Ariadne. The results demonstrate that providing fine-grained provenance via optimized operator instrumentation has minor overhead and clearly outperforms query rewrite, the current state-of-the-art. "

Table 1 provides a comprehensive overview of some presented techniques on data provenance.

Table 1. Summary of Data Provenance Techniques.

System	Course/ Fine grained	Data Granularity	Data Status	Provenance Computing	Provenance Semantics	Storage strategy	Application
(LTDW)[3]	Fine	Tuple	Certain	Lazy	What,Where How	Tightly coupled	Data Warehouse
(ULDB)[4]	Fine	Attribute	Uncertain	Eager	What,Where How	Loosely coupled	Probabilistic DB
(PMCD)[1]	Fine	Attribute	Certain	Eager	What,Where Who,How	Loosely coupled	Curated DB
(PROMS)[5]	Both	All	Certain	Lazy	All	Decoupled	All
(ProQL)[6]	Fine	Tuple	Certain	Lazy	What,Where How	Loosely coupled	CDSSs
(ExSPAN)[7]	Fine	All	Certain	Both	What,How Where	Tightly coupled	Networks
(Panda) [8]	Both	Tuple	Certain	Both	What,Where How,Why	---	Workflow
(Ariadne)[9]	Fine	Tuple	Certain	Both	What, Where, How	Tightly coupled	Networks
(ESvOI)[10]	Fine	Tuple	Certain	Both	What, Where	Tightly coupled	Networks

4. Conclusions And Suggestions For Future Works

In this paper, some data provenance techniques are for different database applications and categorized existing approaches are presented for understanding the current challenges.

As shown in our taxonomy of techniques on data provenance, we classify and analyze present provenance techniques based on our proposed categories, which are provenance granularity, data granularity, data status, provenance computing, semantic of provenance, provenance storage, and applications. From the analyzed research work, we can observe that most research work is proposed for specific application. The provenance information collected by present technique didn't encode all related provenance metadata, which is useful for provenance querying based application. There still remains a lot of research work on querying provenance, which may improve the derived data quality.

The suggestions for future works are presenting generic provenance management techniques, encode more provenance related metadata, and utilize querying provenance for improving data quality.

5. References

Agrawal P., Benjelloun O., Sarma A., Hayworth C., Nabar S., Sugihara T. and Widom J., "Trio: A system for data, uncertainty, and Lineage", in VLDB ,To be in Managing and Mining Uncertain Data, pp. 1151--1154, 2006

Antova L., Koch C., Olteanu D., "Efficient Representation and Processing of Incomplete Information", ICDE, pp. 606-615, 2007.

Benjelloun O., Das Sarma A., Hayworth C., and Widom J., "An introduction to ULDBs and the Trio System", IEEE Data Engineering Bulletin, Special Issue on Probabilistic Databases, vol. 29, no. 1, pp.5–16, 2006.

Buneman P., Chapman v and Cheney J., "Provenance management in curated databases". In SIGMOD, pp.539--550, 2006.

Buneman P., Cheney J., Tan W. C., summeren S. V., "Curated databases", *PODS*, pp.1-12, 2008.

Bunge M., "Treatise on Basic Philosophy", Vol. 3, Ontology I: The Furniture of the World. Boston, MA: Reidel, 1977

Cavallo R., Pittarelli M., "The theory of probabilistic databases", Proc. 13th Internat. Conf. Very Large Databases, pp. 71-81, 1987.

Celko J., "Joe Celko's Data and Databases: Concepts in Practice", Morgan Kaufmann Publishers © 1999.

Chaudhuri S. and Dayal U., "An overview of data warehousing and OLAP technology", SIGMOD Record, vol. 26, no.1, pp. 65–74, 1997.

Chen P. P., "Data Base Management: The E-R Approach to Logical Data Base Design", Wellesley, Mass. Q.E.D. Information Sciences,1977.

Cheney J., "Provenance, XML and the Scientific Web", In ACM SIGPLANWorkshop on Programming Language Technology and XML (PLAN-X 2009), Volume 1, Issue 4, 2009ز

Cheng R., Singh S., Prabhakar S., Shah R., Vitter J. and Xia Y., "Efficient join processing over uncertain data", In Proceedings of ACM 15th Conference on Information and Knowledge Management (CIKM), pp. 738--747,2006

Codd E.F., "A Relational Model of Data for Large Shared Data Banks", Communications of the *ACM* 13 ,no.6, pp. 377–387, 1970ز

Cui Y. and Widom J., "Lineage tracing for general data warehouse transformations", in VLDB Journal, vol.12, Issue 1, pp. 41-58,2003.

Cui Y., Widom J., "Practical Lineage Tracing in Data Warehouses", ICDE, pp.367-378, 2000.

Dalvi N. N., Suciu D., "Efficient query evaluation on probabilistic databases", VLDB J., vol.16, no. 4, pp. 523-544, 2007.

Davies P. B., "Database Systems", Third Edition, published by palgrave MacMillan, 2004

Gennadievna Gagarina, L. ; A. Igorevna Kononova, Y. Olegovna Teplova, A. Vladslavovich Gorodilov, A. Mikhailovich Bain, O. Ivanovich Lisov, V. Georgievich Dorogov and C. Georgievna Dorogova "Development of database design technique as part of the system of ecological monitoring of urban infrastructure objects", Life Science Journal , Vol. 11, No. 11, 2014.

Glavic B., Dittrich K., "Data Provenance: A Categorization of Existing Approaches", In BTW'07, pp. 227—241, 2007.

Glavic, B.; K.S. Esmaili, P. M. Fischer and N. Tatbul" Efficient Stream Provenance via Operator Instrumentation", journal, ACM Transactions on Internet Technology (TOIT), V. 14 Issue 1, Article No. 7, July 2014.

Glavic, B.; K.S. Esmaili, P. M. Fischer and N. Tatbul. Ariadne: Managing fine-grained provenance on data streams. In DEBS, pages 39–50, 2013.

Ikeda R. and Widom J., "Panda: A System for Provenance and Data", Technical Report. Stanford InfoLab,2009.

Inmon W. H., "Tech Topic: What is a Data Warehouse? Prism Solutions", Volume 1. 1995.

Karvounarakis G., Ives Z.G. and Tannen V., "Querying Data Provenance", In Proceedings of the 2010 international conference on Management of data (2010), pp. 951-962, 2010.

Khemaies Boucetta, S. ; D. Daman and S. Shaik" Intelligent selection technique for database indexing to augment the speed performance of query processing on mobile device", Life Science Journal , Vol. 11, No. 4, 2014.

Lomet D. and Widom J., "editors. *Special Issue on Materialized Views and Data Warehousing",* IEEE Data Engineering Bulletin, vol. 18, no. 2, 1995.

Merriam-Webster Web site: http://www.merriam-webster.com/

Ram S. and Liu J., "Understanding the Semantics of Data Provenance to Support Active Conceptual Modeling" , Proceedings of the Active Conceptual Modeling of Learning Workshop (ACM-L 2006) in conjunction with the 25 th International Conference on Conceptual Modeling (ER 2006), vol. 4521, pp 17-29, Springer-Verlag, 2007.

Ram S., Liu J., and George R. T., "PROMS: A system for harvesting and managing data provenance", In WITS, 2006. url: http://kartik.eller.arizona.edu/WITS_DEMO_final.pdf

Scientific Databases, Uni. of South Carolina, Web Site: http://www.sc.edu/library/science/scisdbdb.html.

Singh S.," Database Support for Uncertain Data", Ph.D. Thesis,Center for Education and Research Information Assurance and Security, Purdue University, 2009.

Tan W. C., "Provenance in Databases: Past, Current, and Future", In IEEE Data Eng. Bull. vol.30, no. 4, pp. 3-12, 2007ز

Tan W., "Research Problems in Data Provenance", IEEE Data Engineering Bulletin, vol. 27, no. 4, pp. 42–52, 2004.

Widom J., "Trio: A system for integrated management of data, accuracy and lineage". In Proc. of Conference on Innovative Data System Research (CIDR), 2005.

Widom J., "TRIO:A System for Data, Uncertainty and Lineage", Chapter 1, Dept. of Computer Science, Stanford University,2008.

Zhou W., Sherr M., Tao T., Li X., Loo B.T. and Mao Y., "Efficient Querying and Maintenance of Network Provenance at Internet-Scale", SIGMOD '10 Proceedings of the 2010 international conference on Management of data, 2010