Data Provenance

Origins, Applications, Algorithms, and Models

Other titles in Foundations and Trends® in Databases

Algorithmic Aspects of Parallel Data Processing
Paraschos Koutris, Semih Salihoglu and Dan Suciu
ISBN: 978-1-68083-406-2

Data Infrastructure for Medical Research
Thomas Heinis and Anastasia Ailamaki
ISBN: 978-1-68083-348-5

Main Memory Database Systems
Franz Faerber, Alfons Kemper, Per-Ake Larson, Justin Levandoski, Thomas Neumann and Andrew Pavlo
ISBN: 978-1-68083-324-9

Query Processing on Probabilistic Data: A Survey
Guy Van den Broeck and Dan Suciu
ISBN: 978-1-68083-314-0

Big Graph Analytics Platforms
Da Yan, Yingyi Bu, Yuanyuan Tian and Amol Deshpande
978-1-68083-242-6

Data Provenance

Origins, Applications, Algorithms, and Models

Boris Glavic
Illinois Institute of Technology
bglavic@iit.edu

now
the essence of knowledge
Boston — Delft

Foundations and Trends® in Databases

Published, sold and distributed by:
now Publishers Inc.
PO Box 1024
Hanover, MA 02339
United States
Tel. +1-781-985-4510
www.nowpublishers.com
sales@nowpublishers.com

Outside North America:
now Publishers Inc.
PO Box 179
2600 AD Delft
The Netherlands
Tel. +31-6-51115274

The preferred citation for this publication is

B. Glavic. *Data Provenance*. Foundations and Trends® in Databases, vol. 9, no. 3-4, pp. 209–441, 2021.

ISBN: 978-1-68083-829-9

Foundations and Trends® in Databases
Volume 9, Issue 3-4, 2021

Editorial Board

Editorial Scope

Topics

Foundations and Trends® in Databases publishes survey and tutorial articles in the following topics:

- Data Models and Query Languages
- Query Processing and Optimization
- Storage, Access Methods, and Indexing
- Transaction Management, Concurrency Control and Recovery
- Deductive Databases
- Parallel and Distributed Database Systems
- Database Design and Tuning
- Metadata Management
- Object Management
- Trigger Processing and Active Databases
- Data Mining and OLAP
- Approximate and Interactive Query Processing
- Data Warehousing
- Adaptive Query Processing
- Data Stream Management
- Search and Query Integration
- XML and Semi-Structured Data
- Web Services and Middleware
- Data Integration and Exchange
- Private and Secure Data Management
- Peer-to-Peer, Sensornet and Mobile Data Management
- Scientific and Spatial Data Management
- Data Brokering and Publish/Subscribe
- Data Cleaning and Information Extraction
- Probabilistic Data Management

Information for Librarians

Foundations and Trends® in Databases, 2021, Volume 9, 4 issues. ISSN paper version 1931-7883. ISSN online version 1931-7891. Also available as a combined paper and online subscription.

Contents

Data Provenance

Boris Glavic[1]

[1] *Illinois Institute of Technology; bglavic@iit.edu*

ABSTRACT

Data provenance has evolved from a niche topic to a mainstream area of research in databases and other research communities. This article gives a comprehensive introduction to data provenance. The main focus is on provenance in the context of databases. However, it will be insightful to also consider connections to related research in programming languages, software engineering, semantic web, formal logic, and other communities. The target audience are researchers and practitioners that want to gain a solid understanding of data provenance and the state-of-the-art in this research area. The article only assumes that the reader has a basic understanding of database concepts, but not necessarily any prior exposure to provenance.

Boris Glavic (2021), "Data Provenance", Foundations and Trends® in Databases: Vol. 9, No. 3-4, pp 209–441. DOI: 10.1561/1900000068.

1

Introduction

The term **provenance** is used in the art world to describe a record of the history of ownership of a piece of art. This term has been adapted by the database community to describe a record of the origin of a piece of data. Data provenance has emerged as a research topic in database community in the late 1990's with some isolated earlier work, e.g., Stonebraker *et al.*, 1993. However, as we will discuss in detail in Chapter 5 other research communities have studied concepts that are closely related to data provenance much earlier. Data provenance, by explaining how the result of an operation was derived from its inputs, has proven to be a useful tool that is applicable in a wide variety of applications including debugging transformations (queries, updates, transactions, ...) and data, to assess the trustworthiness of data, to aid users in understanding data-intensive processes, to speed-up incremental maintenance of query results, for explaining surprising outcomes, and for reasoning about hypothetical changes to inputs and results of operations. The purpose of this article is to give a comprehensive introduction to data provenance concepts, algorithms, and methodology developed by the database community in the last few decades. The indented audience are researcher and practitioners unfamiliar with the topic that want to

develop a basic understanding of provenance techniques and the state-of-the-art in the field as well as researchers with some prior experience in provenance that want to broaden their horizon. While also providing a collection of relevant literature references, this article's main objective is to introduce the reader to the formalisms, algorithms, and system's developments in this fascinating field. To be able to cover topics in sufficient depth, we will focus on work on provenance in databases and closely related areas. Provenance for workflow systems, operating systems, general purpose programming languages, and other areas that are not related to databases will not be discussed in depth. That being said, we will point the reader to important work from outside of the database community where appropriate.

1.1 What is Data Provenance?

Following common terminology we will use the term **data item** to refer to a piece of data, e.g., a relation, a tuple, or JSON document. Data items may be inputs and/or outputs of **transformations**, e.g., queries, updates, transactions, application programs. The provenance of a data item provides a record of how the data item was derived from other data items by a set of transformations. We distinguish between **data dependencies** which record that a data item was produced from / depends on another data item, e.g., a tuple in the result of a query was derived from an input tuple, and **transformation dependencies** which record that a data item was directly (or indirectly) produced by a particular transformation. Some provenance models are only concerned with data dependencies or transformation dependencies while others support both types of dependencies. Additional metadata about the execution of a transformation, e.g., the user that executed a transformation or the execution environment of a transformation, are sometimes also considered to be data provenance. However, in this article the main focus is on data and transformation dependencies.

Data and transformation dependencies can be modeled at different levels of **granularity**. For instance, we may track the data dependencies for a query result at the level of attribute values, tuples, or whole relations. The same applies to transformation dependencies, e.g., we

Query result

name	id
Aishe	r_1
Peter	r_2
Astrid	r_3

$Q_{coffee-drinker}$

```
SELECT name FROM student WHERE daily-coffee > 1
UNION
SELECT name FROM teacher WHERE daily-coffee > 1
```

student

name	GPA	daily-coffee	id
Aishe	3.5	2	s_1
James	2.4	0	s_2
Peter	3.6	3	s_3

teacher

name	salary	daily-coffee	id
Alice	30,000	1	t_1
Peter	131,000	2	t_2
Astrid	140,000	3	t_3

Figure 1.1: Example database

may track them at the level of queries or individual operators within a query.

1.1.1 An Example

Consider the database shown in Figure 1.1. The query shown in this figure returns the names of students and teachers (relations student and teacher) who consume more than one cup of coffee per day. Note that this query does not return duplicates (it uses SQL's `UNION` which eliminates duplicates). Let us reason about data dependencies at tuple granularity. That is, we want to know which of the input tuples were used to derive an output tuple. Consider the result $r_1 = (Aishe)$. Student Aishe is in the result, because she drinks more than one cup of coffee per day. The input tuple that justifies the existence of r_1 in the query result is $s_1 = (Aishe, 3.5, 2)$. All other input tuples have no bearing on whether Aishe is in the result or not. Thus, r_1 is only data-dependent on s_1, but no other input tuple. Since Aishe is a student and there is no teacher called Aishe (let alone a teacher drinking a sufficient amount of coffee), the second part of the query that accesses relation

teacher is not needed for producing result tuple r_1. Thus, r_1 is only transformation-dependent on the part of the query highlighted in red.[1] As another example, consider the second result tuple ($r_2 = (Peter)$). There exist two coffee drinkers named Peter. One is a student (tuple s_3) and one is a teacher (tuple t_2). The result tuple r_2 is data-dependent on both of these inputs. Modeling provenance as data dependencies only may not provide us with enough information for all use-cases of data provenance. For example, provenance can help us to determine the effect of deleting a tuple, say s_3, from the input. We can use data dependencies to determine the subset of the output tuples that may be affected by the deletion. Only tuples that are data-dependent on deleted inputs may be affected. However, we need additional information about *how* input tuples were combined by the query to know with certainty whether an output will be deleted or not. For instance, to know whether the deletion of s_3 will cause r_2 to be deleted from the query result, it is not enough to just know the data dependencies of r_2 (which are s_3 and t_2). Additionally, we need to know that as long as one of s_3 or t_3 is in the input, then r_2 will be in the query result. So far we have reasoned intuitively about dependencies. To be able to determine dependencies automatically, we need a formal model of provenance. In Chapter 2 we will review such models and discuss which models provide sufficient information to support particular use-cases in Chapter 3.

1.2 Why Should I Care?

Data provenance has been applied for a diverse set of use cases, many of which we will discuss in detail in Chapter 3. Here we just provide a brief overview of some common use cases.

1.2.1 Error Diagnosis and Debugging

By tracking which input data and parts of a transformation are responsible for producing a suspicious output, provenance information

[1] We may argue about whether the union operation should be considered as relevant or not. For now let us just assume that it is relevant. We will discuss transformation dependencies in detail in Section 2.6.

can be used to narrow down the location of an error. Intuitively, data that was not needed to produce a query result (on which the result is not data dependent on) cannot effect the correctness of the query result. The same applies for transformation dependencies: any part of the transformation that is not a transformation dependency of the result cannot have affected the result.

Example 1 (Debugging)**.** For instance, continuing with the example from Figure 1.1, if the user issuing the query knows that Aishe (tuple r_1) is not a coffee drinker and, thus, should not be in the result of query $Q_{coffee-drinker}$, then the user can restrict their search for errors in the input data to the data dependencies of r_1. In this case the user only needs to inspect s_1 to determine that Aishe's daily-coffee value should 0 instead of having to inspect all six input tuples. In addition to errors in the input, errors in the query result may be caused by bugs in the query. Analog to how data dependencies are used to trace errors to the input data, transformation dependencies can be used to focus attention on the parts of the query (transformation) that could have caused the error. For instance, in our example, only the highlighted part of $Q_{coffee-drinker}$ is responsible for producing result tuple r_1. Obviously, the use of provenance is overkill for this toy example. However, for realistically sized data sets and more complex queries, data and transformation dependencies can significantly improve a user's productivity when debugging data.

1.2.2 Explaining Outcomes

To interpret the result of a complex query, a user may have to understand why and how the result was proved. Data provenance provides such an explanation. However, for large datasets, the full provenance of a query result may be too large to be of any use to a human. Summarization techniques for provenance that produce compact, but semantically meaningful summaries of provenance can be used to address this problem.

Example 2 (Explanations)**.** Continuing with our running example, let us compute the number of non-casual coffee drinkers (more than one cup per day) using the SQL query shown below.

```
SELECT count(*) as num-drinker
FROM (SELECT name, daily-coffee FROM student
      UNION ALL
      SELECT name, daily-coffee FROM teacher) drinkers
WHERE daily-coffee > 1
```

For our example database instance, we get back $r = (4)$, i.e., there are 4 non-casual coffee drinkers. We may explain this result by listing all data dependencies of r: s_1, s_3, t_2, and t_3. However, for larger datasets, the set of data dependencies may be too large to be of any immediate use. Instead, we can employ summarization techniques to compactly describe the set of inputs that are data dependencies. A common summarization technique uses declarative patterns (a limited form of queries) to describe sets of tuples. For instance, instead of listing all data dependencies we may describe them as follows: all students with a GPA higher than 3.4 and all teachers earning more than $100,000$ are non-casual coffee drinkers.

We will discuss summarization techniques for provenance in Section 4.1.

1.2.3 Security and Auditing

Provenance has also been studied extensively by the security community. The record of the operations of a system provided by provenance can be used during the forensic analysis of an attack to understand how a system was breached (Bates and Hassan, 2019). For example, assuming that we collect provenance for every SQL operation executed by a database, then when a user account is compromised this enables us to answer important questions such as *"Which data was accessed or modified by the compromised account?"*. A significant amount of work from the system's security community has investigated how to collect provenance information at the operating system level (Bates *et al.*, 2015; Pasquier *et al.*, 2017). Another security application of data provenance is detection of advanced persistent threads (APTs). It was conjectured it is possible to detect APTs by mining unusual patterns from the provenance of a system. Another security application of provenance is auditing. Many organization have to comply with laws that require

them to report how they processed their data. To support auditing, database systems maintain a record of SQL operations executed in the past available as a so-called *audit-log* (Kaushik *et al.*, 2013; Fabbri and LeFevre, 2011; Kaushik and Ramamurthy, 2011; Agrawal *et al.*, 2004). One disadvantage of audit logs is that they do not record which data was affected / accessed by which operation in the audit log. Provenance information can complement audit logs with data and transformation dependencies to provide this information. Provenance has also been used for access control, e.g., to restrict access to data based on which other data it was derived from (Park *et al.*, 2012).

Example 3 (Enhancing Auditing with Provenance)**.** Consider the scenario shown in Figure 1.2. Bob, a DBA for a bank, has abused his privileges to fix the negative balance of the accounts he has with his employer. Bob has issued two statements (with ids 1432 and 1433) which add $10,000 to both his checkings and savings accounts. The bank identifies account owners by their SSN. Bob's SSN is 333-233-4534. Bob's second statement then reduces the balance of all accounts to compensate for the $20,000 he added to his accounts. To obfuscate his activity, Bob did not select his accounts using his social security number, but instead identified an alternative way to uniquely identify his accounts using their balance, his state, and his age.

The bank maintains an audit log to have a record of all data modifications in case of a security breach or to investigate illegal data access by employees. Bob's illegal operations are recorded in the audit log. However, without knowing the data dependencies and transformation dependencies (the provenance) of these operations, it is not obvious what the purpose and effect of the illegal operations were. Transformation dependencies would unearth that the first statement only affected Bob's account. Data dependencies between tuples in the database state before the updates and after the updates can be used to determine how Bob's changes can be undone.

Of course, Example 3 is too simple to be realistic. Nonetheless, it illustrates how provenance can complement audit logging. In Section 2.7 we will introduce provenance models for update operations and transactions. One major challenges with supporting provenance for updates

Database state before statement 1432

Owner	State	Age	Balance	Type
333-233-4534	IL	36	-3,030	Checking
333-233-4534	IL	36	-1,000	Savings
111-232-2323	IL	34	100,004	Checking
...	...	...	...	...

Database state after statement 1432

Owner	State	Age	Balance	Type
333-233-4534	IL	36	6,970	Checking
333-233-4534	IL	36	9,000	Savings
111-232-2323	IL	34	100,004	Checking
...	...	...	...	...

Database state after statement 1433

Owner	State	Age	Balance	Type
333-233-4534	IL	36	6,969.99	Checking
333-233-4534	IL	36	8,999.99	Savings
111-232-2323	IL	34	100,003.99	Checking
...	...	...	...	...

Audit log

id	acc	timestamp	statement
...	...	...	...
1432	Bob	01-01 8:34	`UPDATE accounts SET Balance = Balance + 10,000 WHERE state = 'IL' AND Age = 36 AND balance IN (-3,030, -1,000)`
1433	Bob	01-01 8:35	`UPDATE accounts SET Balance = Balance - (20000.0 / (SELECT count(*)) FROM accounts))`
...	...	...	...

Figure 1.2: Example audit log and database states

is that it requires access to past database versions, i.e., tuples in the database state after the update are data-dependent on tuples in the database state before the update. We will discuss provenance models for updates in Section 2.7 and methods for capturing update provenance in Section 4.3.

1.2.4 View Maintenance and Provisioning

Many provenance models use a symbolic representation of a computation to record how inputs have been combined by a transformation. Such representations are often invariant under certain changes to the input (and/or the transformation). That is, it is possible to use the provenance of a query result to determine how this result would be affected when the input or transformation is changed in a certain way. A reader familiar with the literature may recognize this is as the well-known **view maintenance** problem (Gupta, Mumick, *et al.*, 1995): given a database D, a query Q, the query's result $Q(D)$ and an update ΔD, compute $Q(D \cup \Delta D)$. Let use consider deletion propagation for a simple projection query as an example of how to exploit provenance information for view maintenance. For sake of the example, we will use a simple provenance model that records the data dependencies for a query result tuple at tuple-granularity. That is, the provenance for each output tuple of the query is the set of input tuples it depends on.

Example 4 (View Maintenance)**.** Consider the SQL query shown below which returns customers which have ordered at least one item. This query returns two tuples: $(Peter)$ whose provenance consists of the three tuples $\{t_1, t_2, t_3\}$ (all the orders from Peter) and $(Alice)$ whose provenance is $\{t_4\}$ (all of her orders). To determine the effect of deleting some of the input tuples, we can simply remove them from the provenance of the result tuples. Any tuple whose provenance is empty will no longer be in the result. For example, if we delete t_1 and t_2 then tuple $(Peter)$ is still in result, because its provenance is not empty ($\{t_3\}$), because one of Peter's order is still present in the input.

```
SELECT DISTINCT customer FROM orders
```

customer	item	id
Peter	Umbrella	t_1
Peter	Raincoat	t_2
Peter	Gumboots	t_3
Alice	Umbrella	t_4

We will see in Chapter 2 and Section 3.1 that in general we will need provenance models that record how input data was combined by a computation to be able to use provenance effectively for view maintenance. A specific type of view maintenance is what-if analysis where a user wants to evaluate the effect a hypothetical change to their data has on a query result. Provenance information can be used to provision for what-if analysis if the what-if analysis is restricted to set of scenarios that are known upfront (Assadi *et al.*, 2016).

1.2.5 View Update and How-to Analysis

In the **view update** problem we are given a query Q and database D and an update $\Delta Q(D)$ to the query's result $Q(D)$ as input and have to translated this update into an update ΔD of the database such that $Q(D \cup \Delta D) = Q(D) \cup \Delta Q(D)$. Since such a ΔD may not exist for all inputs, the problem is often relaxed to allow for side-effects, i.e., the query result over $D \cup \Delta D$ is not exactly $Q(D) \cup \Delta Q(D)$. The view update problem is typically stated as an optimization problem where the goal is to find a delta ΔD that minimizes side-effects on the input database and/or query result. Provenance information aids in view update by identifying which inputs needs to be modified to achieve a desired update to a query's result.

Example 5 (View Update)**.** Continuing with Example 4, assume we would like to delete Peter from the query result. This can be achieved by deleting all of the tuples from the input database that the result tuple $(Peter)$ depends on (his orders). In this example, these are input tuples t_1, t_2, and t_3.

Closely related to the view update problem are how-to queries (Meliou and Suciu, 2012) where constraints on what is a desired query result are specified declaratively and the goal is to produce an update

Loan applications

name	loanamount	income	assets	criminalrecord
Peter	10,000	15,000	500	1
Alice	50,000	135,000	10,000	0
Bob	35,000	20,000	200,000	0

Normalized data

name	loanamount	income	assets	criminalrecord
Peter	0.1	0.05	0.002	1.0
Alice	0.5	0.45	0.04	0.0
Bob	0.35	0.06	0.8	0.0

Linear classifier

$$\mathcal{C}(t) = -0.2 \cdot t.loanamount + 0.6 \cdot t.assets - 0.3 \cdot t.criminalrecord$$

Figure 1.3: Explaining classification with provenance

to the input database such that the user's constraints are fulfilled and a used-specified optimization goal is met. We will discuss applications of provenance to view update problems in Section 3.2.

1.2.6 Explaining Machine Learning Models and Outcomes

Transparency, fairness, and explainability in machine learning are of immense importance, because decisions that have significant real world impact, e.g., whether to accept or reject loan applications or whether to hire an applicant, are often delegated to machine learning models. Provenance information, while not a magical solution for explainability, is certainly relevant for explaining the outcome of applying an ML model to classify an input as well as for explaining how the training data and algorithm used to train the model affected the outcome indirectly by determining the model.

Example 6 (Explaining ML outcomes)**.** Consider a bank that has uses a linear classifier to decide loan applications. An example relation and the model are shown in Figure 1.3. Loan applications for which the model $\mathcal{C}$ returns a negative number are denied and loan applications where $\mathcal{C}$ is

positive are granted. Note that the income of a person is not considered when making loan decisions. Data dependencies at the granularity of attribute values can help us identify which features of a loan application are considered in the loan application decision process. However, the the degree to which a feature affects the result depends on the feature. Note that this is common for machine learning models to have all (or most) features affect the classification outcome, but not all to the same degree. Data dependencies only record the existence of an a dependency, but do not measure the amount of influence. For instance, in our example linear model, features assets and criminal-record have large weights, i.e., have larger impact on the result in general than loan-amount and income. However, whether a feature can be held responsible for the classification of an input o also depends on o's features. This is less of a concern with simple models (like our linear classifier), but can be significant for models where features are not treated independently.

In Section 2.1.5 we will review the notion of responsibility which measures the degree of impact an input has on the output of a query. We will discuss the relationship of provenance and other types of explanations for ML models and outcomes in Section 1.2.6.

1.3 Background and Notation

We now introduce notational conventions used in this article and briefly review the relational data model and some relational query languages. Readers familiar with these concepts may skip this section.

1.3.1 The Relational Model

A relation schema $\mathbf{R}$ is a list of attribute names $(A_1, \ldots, A_n)$. The arity $arity(\mathbf{R})$ of a relation schema $\mathbf{R}$ is the number of attributes in the schema. Consider a universal domain $\mathcal{U}$ of values. Under the *named perspective* a tuple over a relation schema $\mathbf{R}$ is a function that maps attributes from $\mathbf{R}$ to values from $\mathcal{U}$. Under the *unnamed perspective* a tuple with arity n is an element from $\mathcal{U}^n$. We will opportunistically switch between these perspectives to simplify the exposition. A relation R of schema $\mathbf{R} = (A_1, \ldots, A_n)$ under set semantics is a set

of tuples over schema $\mathbf{R}$. A database schema $\mathbf{D}$ is a set of relation schema $\{\mathbf{R}_1, \ldots, \mathbf{R}_m\}$. A database D over a schema $\mathbf{D}$ is set of relations $\{R_1, \ldots, R_n\}$, one for each relation schema in $\mathbf{D}$. If R is a relation (D is a database) then we use $\mathbf{R}$ ($\mathbf{D}$) to denote its schema. We will sometimes consider a slightly different definition of relations where each attribute A_i is associated with a domain $\mathcal{D}_i$. Under this model, Relation schemas are lists $(A_1 : \mathcal{D}_1, \ldots, A_n : \mathcal{D}_n)$ and a set relation of schema $(A_1 : \mathcal{D}_1, \ldots, A_n : \mathcal{D}_n)$ is a subset of $\mathcal{D}_1 \times \ldots \times \mathcal{D}_n$.

A bag semantics (or multiset) relation R of arity n is a multiset of tuples from $\mathcal{U}^n$. That is, in R each tuple is associated with a multiplicity (the number of duplicates of the tuple that are in the relation). Formally, a bag relation is a function $R : \mathcal{U}^n \to \mathbb{N}$ that associates each tuple $t \in \mathcal{U}^n$ with natural number $R(t)$ (its multiplicity). Note that here we consider bag relations to be total functions. Tuples that do not exist in the relation are assigned multiplicity 0. If $\mathcal{U}$ is infinite, then we require that there exist only finitely many t such that $R(t) \neq 0$. That is, we only consider finite relations. The use of total functions may seem to complicate matters unnecessarily, but will be beneficial when we discuss provenance models in Chapter 2.

As we use the two query languages, relational algebra and Datalog, extensively throughout this work, we briefly review them below.

1.3.2 Relational Algebra

Given a query Q, we use $\text{SCH}(Q)$ to denote the schema of the result of Q. We use $[\![Q]\!]_D$ to denote the result of evaluating query Q over database D. The arity $arity(Q)$ of a query Q is the arity of $\text{SCH}(Q)$. Sometimes we will use $Q(D)$ instead of $[\![Q]\!]_D$. For set semantics relations we will use the relational algebra shown in Figure 1.4.

As is customary, we define the semantics of algebra operators using set compressions. We use $\circ$ to denot concatenation of tuples (and other types of sequences). For a tuple t, $t.A$ denotes the projection of the tuple onto a list of expressions. A relation access R returns the instance of this relation in database D. Selection returns all input tuples t that fulfill a condition θ, written as $t \models \theta$. Projection projects all input tuples onto a list of expressions A. We typically will assume that

$$
\begin{aligned}
[\![R]\!]_D &= R && \textbf{(Relation access)}\\
[\![\sigma_\theta(Q)]\!]_D &= \{t \mid t \in [\![Q]\!]_D \wedge t \models \theta\} && \textbf{(Selection)}\\
[\![\Pi_A(Q)]\!]_D &= \{t.A \mid t \in [\![Q]\!]_D\} && \textbf{(Projection)}\\
[\![\rho_{A\to B}(Q)]\!]_D &= \{t[A \to B] \mid t \in [\![Q]\!]_D\} && \textbf{(Renaming)}\\
[\![Q_1 \times Q_2]\!]_D &= \{t_1 \circ t_2 \mid t_1 \in [\![Q_1]\!]_D \wedge t_2 \in [\![Q_2]\!]_D\} && \textbf{(Crossproduct)}\\
[\![Q_1 \cup Q_2]\!]_D &= [\![Q_1]\!]_D \cup [\![Q_2]\!]_D && \textbf{(Union)}\\
[\![Q_1 \cap Q_2]\!]_D &= [\![Q_1]\!]_D \cap [\![Q_2]\!]_D && \textbf{(Intersection)}\\
[\![Q_1 - Q_2]\!]_D &= [\![Q_1]\!]_D - [\![Q_2]\!]_D && \textbf{(Difference)}\\
[\![\gamma_{f(A)\to B}(Q)]\!]_D &= \{(f([\![\Pi_A(Q)]\!]_D)\} && \textbf{(Aggregation)}\\
[\![\gamma_{f(A)\to B;G}(Q)]\!]_D &= \{f([\![\Pi_A(\sigma_{G=t.G}(Q))]\!]_D) \circ g \mid g \in [\![\Pi_G(Q)]\!]_D\}
\end{aligned}
$$

Figure 1.4: Set semantics relational algebra

such expressions can consist of references to attributes, constants, and arithmetic operations, e.g., $(A+3)*C$. This variant of projection if often referred to as generalized projection. In contexts where A is subject to restrictions, we will explicitly mention that. For convenience, we will also allow projection to rename the results of expression, e.g. ,$\Pi_{A+B\to C}$ projects the input on $A + B$ and the attribute storing the result of this expression is named C. Renaming $\rho_{A_1\to B_1,\ldots,A_n\to B_n}$ renames attribute A_i as B_i. As a notational shortcut we will write $A \to B$ were both $A = (A_1, \ldots A_n)$ and $B = (B_1, \ldots, B_n)$ are lists of attributes to denote $A_1 \to B_1, \ldots, A_n \to B_n$. Here, $t[A \to B]$ denotes renaming attributes in the schema of tuple t (assuming the named perspective). Cross product is the set-theoretical cross product of the two input relations. Join $Q_1 \bowtie Q_2$ (not shown in the figure) is syntatic sugar for a cross product followed by a selection. Set operations (union, intersection, and difference) are defined as in set theory. These operations are only defined for inputs of the same arity (with the same data types if we consider typed relations). We consider two variants of aggregation: aggregation with group-by and aggregration without group-by. Aggregation without

group-by applies an aggregation function $f : \{\mathcal{U}\} \to \mathcal{U}$ to all values from an attribute A. The result is a relation with a single tuple and single attribute named B. Note that this operator returns a single result tuple, even if the input is empty. Aggregation with group-by, partitions the input relation into groups (subsets) such that each group consists of precisely the set of tuples that have a particular value in the group-by attributes G. The aggregation function f is then applied to each group. The operator returns a tuple for each group consisting of the aggregation function result for that group and the group-by attribute values. Throughout this paper we will also allow aggregation to apply a (possibly empty) list of aggregation functions instead of a single aggregation, e.g., $\gamma_{count(*),sum(salary);dept}(\mathsf{employee})$.

Example 7 (Relational algebra (set semantics))**.** The SQL queries from Figure 1.1 and Example 2 can be written relational algebra as:

$$\Pi_{name}(\mathsf{student}) \cup \Pi_{name}(\mathsf{teaching}) \quad \text{(Figure 1.1)}$$

$$\gamma_{count(*)}(\sigma_{daily-coffee>1}(\Pi_{name,daily-coffee}(\mathsf{student}) \cup \Pi_{name,daily-coffee}(\mathsf{teaching}))) \quad \text{(Example 2)}$$

To be able to reason about SQL databases which use the bag semantics version of the relational model, we also introduce a bag semantics version of relational algebra. The semantics of the operators of this algebra is defined in Figure 1.5. Recall that we model bag relations as functions from tuples to the set of natural numbers $\mathbb{N}$. Thus, a query result is a function that maps result tuples to their multiplicity. In Figure 1.5 we define these functions pointwise, i.e., we define how to calculate the multiplicity of a query result tuple t (the result of applying function $[\![Q]\!]_D$ to t) based on the multiplicities of input tuples.[2] For instance, the number of duplicates of a tuple t in the result of $Q_1 \cup Q_2$ is the sum of the number of duplicates of t in the result of Q_1 and in the result of Q_2. Projection sums up the multiplicities of all input tuples that are projected onto the result tuple t. For a cross product we have to multiply the multiplities of input tuples. The set

[2] A reader familiar with $\mathcal{K}$-relations may recognize that for positive relational algbra we have defined the semantics of operators as is done for $\mathcal{K}$-relations. This is deliberate and will come in handy when we discuss $\mathcal{K}$-relations in Chapter 2.

$$[\![R]\!]_D(t) = R(t) \qquad \textbf{(Relation access)}$$

$$[\![\sigma_\theta(Q)]\!]_D(t) = \begin{cases} [\![Q]\!]_D(t) & \text{if } t \models \theta \\ 0 & \text{otherwise} \end{cases} \qquad \textbf{(Selection)}$$

$$[\![\Pi_A(Q)]\!]_D(t) = \sum_{t'.A=t} [\![Q]\!]_D(t') \qquad \textbf{(Projection)}$$

$$[\![\rho_{A\to B}(Q)]\!]_D(t) = [\![Q]\!]_D(t[B \to A]) \qquad \textbf{(Renaming)}$$

$$[\![Q_1 \times Q_2]\!]_D(t) = [\![Q_1]\!]_D(t[\textsc{Sch}(Q_1)]) \cdot [\![Q_2]\!]_D(t[\textsc{Sch}(Q_2)]) \qquad \textbf{(Crossproduct)}$$

$$[\![Q_1 \cup Q_2]\!]_D(t) = [\![Q_1]\!]_D(t) + [\![Q_2]\!]_D(t) \qquad \textbf{(Union)}$$

$$[\![Q_1 \cap Q_2]\!]_D(t) = \min([\![Q_1]\!]_D(t), [\![Q_2]\!]_D(t)) \qquad \textbf{(Intersection)}$$

$$[\![Q_1 - Q_2]\!]_D(t) = \max(0, [\![Q_1]\!]_D(t) - [\![Q_2]\!]_D(t)) \qquad \textbf{(Difference)}$$

$$[\![\gamma_{f(A)\to B}(Q)]\!]_D(t) = \begin{cases} 1 & \textbf{if } t = (f([\![\Pi_A(Q)]\!]_D) \\ 0 & \textbf{otherwise} \end{cases} \qquad \textbf{(Aggregation)}$$

$$[\![\gamma_{f(A)\to B;G}(Q)]\!]_D(t) = \begin{cases} 1 & \textbf{if } t.G \in \{t.G \mid [\![Q]\!]_D(t) > 0\} \\ & \quad \wedge t.B = f([\![\Pi_A(\sigma_{G=t.G}(Q))]\!]_D) \\ 0 & \textbf{otherwise} \end{cases}$$

Figure 1.5: Bag semantics relational algebra

operations, as expected, correspond to SQL's `UNION ALL`, `INTERSECT ALL`, and `EXCEPT ALL`. As syntactic sugar, we will use δ to denote duplicate elimination which can be expressed as group-by aggregation without an aggregation function.

Example 8 (Relational algebra (bag semantics))**.** For instance, below we show the query from Example 4 expressed in bag semantics relational algebra using duplicate elimination and using group-by aggregation.

$$\delta(\Pi_{customer}(\mathsf{orders}))$$
$$\gamma_{customer}(\mathsf{orders})$$

1.3.3 Datalog

Datalog is a query language based on formal logic. A Datalog program consists of a set of rules of the form

$$r : \underbrace{Q(\overline{X})}_{\text{head}} :- \underbrace{\mathsf{R}_1(\overline{X_1}), \ldots, \mathsf{R}_n(\overline{X_n})}_{\text{body}}$$

where Q is a predicate (q relation) and all R_i are predicates or their negation and each $\overline{X_i}$ is a list of variables from an infinite set of variable symbols $\mathcal{V}$ and constants from a domain $\mathcal{U}$. We use $vars(r)$ to denote the set variables that occur in rule r. Each $\mathsf{R}_i(\overline{X_i})$ is called a goal. The left-hand side (LHS) of a Datalog rule r is called its head $head(r)$ and the right-hand side (RHS) is its body $body(r)$. A Datalog rule represents a logical implication:

$$\mathsf{R}_1(\overline{X_1}) \wedge \ldots \wedge \mathsf{R}_n(\overline{X_n}) \rightarrow Q(\overline{X})$$

A Datalog rule is safe if all the variables in the head occur in at least one positive (non-negated) body goal. A Datalog program P consists of a set of Datalog rules. The relations occurring in a Datalog are partitioned into two sets. The extensional database (or *edb*) are relations that do not occur in the head of rules in P, i.e., these are the relations in the database. The intentional database (or *idb*) are relations that occur in the head of rules (the relations computed by the program). The set of *edb* and *idb* relations are required to be disjoint. Note that

in contrast to a relational algebra expression, Datalog programs may compute multiple result relations. A Datalog query Q is a Datalog program with a distinguished *idb* relation Q called the answer relation. Note that Datalog programs can be recursive, e.g., the head predicate of a rule may appear in the rule's body. A canonical example of recursion is the computation of the transitive closure of the edge relation of a graph.

The semantics of a Datalog program can be defined in several equivalent ways. Discussing these different semantics in detail is beyond the scope of this paper. We refer the interested reader to Abiteboul *et al.* (1995) and Ceri *et al.* (1989). For example, the model-theoretic semantics of Datalog treats the Datalog program and extensional database as a set of sentences in first-order logic and defines the result of the program to be the smallest model for this set of sentences.

Here we will use the fixed points semantics of Datalog. A valuation $\varphi : vars(r) \rightarrow \textsc{adom}(D)$ assigns variables from a rule r to constants from the active domain $\textsc{adom}(D)$ of a database D (which we refer to as an *edb* instance in the context of Datalog). The active domain of a database is the set of constants that occur in D. A valuation is applied to a Datalog rule r by replacing each variable X in r with $\varphi(X)$. We refer to $\varphi(r)$ as a *grounded rule*. The semantics of evaluating program P over an *edb* database instance D is the least fix point, denoted as $T_P^*(D)$, of the immediate consequences operator T_P over D. Intuitively, the immediate consequence operator takes as input an instance I and returns all new facts that can be derived based on the facts in I using the rules of P, i.e., that are the heads of grounded rules where the body evaluates too true in I. Note that as mentioned above the body of a Datalog rule is interpreted as a conjunction of its goal. A positive grounded goal $\mathsf{R}(\bar{c})$ evaluates to true over an instance I if $\mathsf{R}(\bar{c})$ exists in I. A negated goal $\neg\, \mathsf{R}(\bar{c})$ evaluates to true over I if $\mathsf{R}(\bar{c}) \notin I$. The immediate consequence operator and its fixed point T_P^* are defined below.

$$
\begin{aligned}
T_P(I) &= \{\varphi(head(r)) \mid r \in P \wedge I \models \varphi(body(r))\} \\
T_P^0(D) &= D \\
T_P^{n+1}(D) &= T_P^n(D) \cup T_P(T_P^n(D)) \\
T_P^*(D) &= T_P^m(D) \text{ where } m = \operatorname*{argmin}_{i \in \mathbb{N}}(T_P^i(D) = T_P^{i+1}(D))
\end{aligned}
$$

Note that the least fixed point $T_P^*(D)$ is guaranteed to exist and to be unique for all positive Datalog programs.

Example 9 (Transitive Closure)**.** The transitive closure of the edge relation of a directed graph contains all pairs of nodes (a, b) such that there exists a path from a to b. Below we show a recursive Datalog program P_{tc} that computes the transitive closure over a relation $\mathsf{edge}(in, out)$ storing the edges of the input graph. Rule r_1 initializes the transitive closure with the end points of all paths of length 1 (the edges of the graph). Rule r_2 takes the end points of a path of length n (a pair of nodes that we already have established to be in the transitive closure) and returns the end points of an extension of such a path by one additional edge. Figure 1.6 shows an example graph and the evaluation of P_{tc} over this graph using the immediate consequence operator.

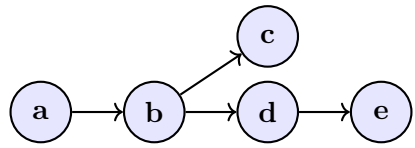
$$
\begin{aligned}
r_1 &: \mathsf{tc}(X, Y) :- \mathsf{edge}(X, Y) \\
r_2 &: \mathsf{tc}(X, Y) :- \mathsf{tc}(X, Z), \mathsf{edge}(Z, Y)
\end{aligned}
$$

1.3.4 Query Classes

Many provenance models are limited to certain classes of queries, e.g., positive relational algebra. Here we review commonly used classes of queries.

Relational Algebra

The full relational algebra $\mathcal{RA}$ consists of operators projection, union, selection, cross product, and difference. Positive relational algebra $\mathcal{RA}^+$



edge

in	out
a	b
b	c
b	d
d	e

I^0

in	out
a	b
b	c
b	d
d	e

I^1

X	Y
a	b
b	c
b	d
d	e
a	c
a	d
b	e

I^2

X	Y
a	b
b	c
b	d
d	e
a	c
a	d
b	e
a	e

Figure 1.6: Computing the transitive closure of a graph with Datalog

consists of all monotone operators of relational algebra, i.e., all operators of $\mathcal{RA}$ expect difference. The name stems from the fact that if expressed in formal logic, queries of $\mathcal{RA}^+$ do not contain negation.[3]. Adding aggregation to full relational algebra, we get the class $\mathcal{RA}^{agg}$.

Datalog and First-Order Logic

Many fundamental classes of queries have a natural representation in Datalog. Conjunctive queries ($\mathcal{CQ}$) are queries that consist of a single Datalog rule without negation and comparison predicates and where all body atoms edb relations. By allowing certain comparison predicates in conjunctive queries we get the classes of conjunctive queries with inequalities $\mathcal{CQ}^{\neq}$ and conjunctive queries with ordering $\mathcal{CQ}^{<}$. A union of conjunctive queries ($\mathcal{UCQ}$) is a Datalog query with one or more rules that are each conjunctive queries. Non-recursive positive Datalog programs may reference *idb* relations in rule bodys,

[3] Assuming that comparison operators such as $\neq$ are "build-in" and not considered as negation, e.g., $a \neq b$ instead of $\neg(a = b)$.

but no direct or indirect recursion or negated atoms are allowed. This class of queries is equivalent in terms of expressive power to the class $\mathcal{UCQ}$, but queries in this class can be exponentially more concise than the corresponding queries from $\mathcal{UCQ}$. First-order queries $\mathcal{FO}$ are non-recursive Datalog programs with negated atoms. The class DATALOG consists of, possibly recursive, rules without negated atoms. DATALOG$^\neg$ is the class of Datalog programs with recursion and negation. Note that the fixed point semantics for Datalog we have introduced is no longer sufficient for dealing with programs that contain both recursion and negation, because such programs may not have a unique smallest model. Several alternative semantics have been proposed in the literature to deal with this, e.g., the well-founded semantics (Van Gelder *et al.*, 1991), the stratified semantics (Chandra and Harel, 1985) which is only applicable to stratified programs (a subclass of DATALOG$^\neg$), and the inflationary semantics (Kolaitis and Papadimitriou, 1988). However, the discussion of these semantics is beyond the scope of this paper.

1.3.5 Query Equivalence and Containment

Semantically, queries can be viewed as functions that map databases (their input) to relations (the query result). It is often important to be able to identify two queries which differ in syntax, but have the same semantics, i.e., produce the same result over all databases. Such queries are called *equivalent*. For certain applications it is useful to generalize this notion and reason about whether one query Q provides strictly more information than another query Q', i.e., for any database query Q' returns a subset of Q's result.

Definition 1 (Query Equivalence and Containment)**.** Given two queries Q and Q' over the same database schema, we say that Q is *equivalent* to Q', written as $Q \equiv Q'$ if:

$$\forall D : Q(D) = Q'(D)$$

Query Q is *contained* in query Q', written as $Q \sqsubseteq Q'$ if:

$$\forall D : Q(D) \subseteq Q'(D)$$

Note that $Q \equiv Q'$ if and only if $Q \sqsubseteq Q' \wedge Q' \sqsubseteq Q$.

1.4 Organization of this Monograph

The chapters of this article were written to be mostly self-contained. That being said, a basic understanding of provenance models is necessary for following the discussion in Chapter 3 and Chapter 4. Thus, we recommend readers without background in formal provenance models to read the beginning of Chapter 2 first before moving on to later chapters.

Chapter 2 introduces the reader to models that define a formal semantics for provenance. We will introduce well-established models and will compare them with respect to their expressive power, correctness guarantees, and supported transformation languages. Furthermore, we will shine light on the relationship between provenance for non-monotone queries and why-not provenance which explains missing answers.

In Chapter 3 we will discuss several applications that benefit from or are enabled by data provenance. As we already hinted at in Section 1.2, provenance can aide in a variety of view maintenance and update problems, is used to debug transformations and data, can serve as the foundation for explanations of outcomes, and is applied to explain predictions and models in machine learning.

In Chapter 4 we will discuss algorithms, techniques, and systems that manage provenance information. Our main focus will be on how to *represent and store* provenance information, how to automatically *capture* provenance information, and how to *query* data provenance.

We will cover research from other communities that is closely to related to data provenance in Chapter 5. These include data- and controlflow analysis, program slicing, and other related program analysis techniques that have been developed by the software engineering, programming languages, and compiler communities; taint analysis that has been used extensively by the security community; justifications and debugging for logic programming; symbolic program execution; and explainability in machine learning.

References

Abiteboul, S., R. Hull, and V. Vianu. (1995). *Foundations of Databases*. Addison-Wesley.

Agrawal, P., O. Benjelloun, A. D. Sarma, C. Hayworth, S. Nabar, T. Sugihara, and J. Widom. (2006). "An Introduction to ULDBs and the Trio System". *IEEE Data Engineering Bulletin*. 29(1): 5–16.

Agrawal, R., R. Bayardo, C. Faloutsos, J. Kiernan, R. Rantzau, and R. Srikant. (2004). "Auditing compliance with a hippocratic database". In: *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*. VLDB Endowment. 516–527.

Akoush, S., R. Sohan, and A. Hopper. (2013). "HadoopProv: Towards Provenance As A First Class Citizen In MapReduce". *TaPP*.

Alexe, B., L. Chiticariu, and W. Tan. (2006). "SPIDER: a schema mapPIng DEbuggeR". In: *Proceedings of the 32nd international conference on Very large data bases*. VLDB Endowment. 1179–1182.

Allen, F. (1970). "Control flow analysis". *Proceedings of a symposium on Compiler optimization*. 5(7): 1–19.

Alvaro, P., J. Rosen, and J. M. Hellerstein. (2015). "Lineage-driven Fault Injection". In: *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. ACM. 331–346.

Amarilli, A., P. Bourhis, and P. Senellart. (2015). "Provenance Circuits for Trees and Treelike Instances". In: *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part II*. 56–68. DOI: 10.1007/978-3-662-47666-6_5. URL: https://doi.org/10.1007/978-3-662-47666-6%5C_5.

Amsterdamer, Y., S. Davidson, D. Deutch, T. Milo, J. Stoyanovich, and V. Tannen. (2011a). "Putting Lipstick on Pig: Enabling Database-style Workflow Provenance". *Proceedings of the VLDB Endowment*. 5(4): 346–357.

Amsterdamer, Y., D. Deutch, T. Milo, and V. Tannen. (2011b). "On Provenance Minimization". In: *Proceedings of the 30th Symposium on Principles of Database Systems (PODS)*. 141–152.

Amsterdamer, Y., D. Deutch, and V. Tannen. (2011c). "On the Limitations of Provenance for Queries with Difference". In: *TaPP '11: 3rd USENIX Workshop on the Theory and Practice of Provenance*.

Amsterdamer, Y., D. Deutch, and V. Tannen. (2011d). "Provenance for Aggregate Queries". In: *Proceedings of the thirtieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. ACM. 153–164.

Anand, M. K., S. Bowers, and B. Ludäscher. (2010). "Techniques for efficiently querying scientific workflow provenance graphs." In: *EDBT*. Vol. 10. 287–298.

Anand, M. K., S. Bowers, T. McPhillips, and B. Ludäscher. (2009). "Efficient Provenance Storage over Nested Data Collections". In: *EDBT '09: Proceedings of the 12th International Conference on Extending Database Technology*. 958–969.

Apache. *http://atlas.apache.org/*. (Accessed on 2017).

Arab, B., S. Feng, B. Glavic, S. Lee, X. Niu, and Q. Zeng. (2018a). "GProM - A Swiss Army Knife for Your Provenance Needs". *IEEE Data Engineering Bulletin*. 41(1): 51–62.

Arab, B., D. Gawlick, V. Krishnaswamy, V. Radhakrishnan, and B. Glavic. (2016). "Reenactment for Read-Committed Snapshot Isolation". In: *Proceedings of the 25th ACM International Conference on Information and Knowledge Management*. 841–850.

Arab, B., D. Gawlick, V. Krishnaswamy, V. Radhakrishnan, and B. Glavic. (2018b). "Using Reenactment to Retroactively Capture Provenance for Transactions". *IEEE Transactions on Knowledge and Data Engineering*. 30(3): 599–612. DOI: 10.1109/TKDE.2017.2769056.

Arocena, P. C., B. Glavic, and R. J. Miller. (2013). "Value Invention for Data Exchange". In: *Proceedings of the 39th International Conference on Management of Data*. 157–168.

Assadi, S., S. Khanna, Y. Li, and V. Tannen. (2016). "Algorithms for Provisioning Queries and Analytics". In: *19th International Conference on Database Theory, ICDT 2016, Bordeaux, France, March 15-18, 2016*. 18:1–18:18. DOI: 10.4230/LIPIcs.ICDT.2016.18. URL: https://doi.org/10.4230/LIPIcs.ICDT.2016.18.

Bakibayev, N., D. Olteanu, and J. Závodný. (2012). "FDB: A query engine for factorised relational databases". *Proceedings of the VLDB Endowment*. 5(11): 1232–1243.

Bao, Z., H. Köhler, L. Wang, X. Zhou, and S. W. Sadiq. (2012). "Efficient provenance storage for relational queries". In: *21st ACM International Conference on Information and Knowledge Management, CIKM'12, Maui, HI, USA, October 29 - November 02, 2012*. Ed. by X. Chen, G. Lebanon, H. Wang, and M. J. Zaki. ACM. 1352–1361. ISBN: 978-1-4503-1156-4. DOI: 10.1145/2396761.2398439. URL: https://doi.org/10.1145/2396761.2398439.

Bates, A. and W. U. Hassan. (2019). "Can Data Provenance Put an End To the Data Breach?" *IEEE Security & Privacy*. 17(4): 88–93. DOI: 10.1109/MSEC.2019.2913693. URL: https://doi.org/10.1109/MSEC.2019.2913693.

Bates, A., D. Tian, K. R. B. Butler, and T. Moyer. (2015). "Trustworthy Whole-System Provenance for the Linux Kernel". In: *24th USENIX Security Symposium, USENIX Security 15, Washington, D.C., USA, August 12-14, 2015*. Ed. by J. Jung and T. Holz. USENIX Association. 319–334. URL: https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/bates.

Becker, R. A. and J. M. Chambers. (1988). "Auditing of data analyses". *Journal on Scientific and Statistical Computation*: 747–760.

Benjelloun, O., A. D. Sarma, A. Y. Halevy, and J. Widom. (2006). "ULDBs: Databases with Uncertainty and Lineage". In: *Proceedings of the 32th International Conference on Very Large Data Bases (VLDB)*. 953–964.

Berenson, H., P. Bernstein, J. Gray, J. Melton, E. O'Neil, and P. O'Neil. (1995). "A critique of ANSI SQL isolation levels". *ACM SIGMOD Record*. 24(2): 1–10.

Bertossi, L. and B. Salimi. (2013). "Causality in Databases, Database Repairs, and Consistency-Based Diagnosis".

Bhagwat, D., L. Chiticariu, W. C. Tan, and G. Vijayvargiya. (2005). "An Annotation Management System for Relational Databases". *VLDB J.* 14(4): 373–396. DOI: 10.1007/s00778-005-0156-6. URL: https://doi.org/10.1007/s00778-005-0156-6.

Bhagwat, D., L. Chiticariu, W.-C. Tan, and G. Vijayvargiya. (2004). "An Annotation Management System for Relational Databases". In: *VLDB '04: Proceedings of the 30th International Conference on Very Large Data Bases*. 900–911.

Bhardwaj, A., A. Deshpande, A. J. Elmore, D. Karger, S. Madden, A. Parameswaran, H. Subramanyam, E. Wu, and R. Zhang. (2015). "Collaborative data analytics with DataHub". *Proceedings of the VLDB Endowment*. 8(12): 1916–1919.

Bhattacherjee, S., A. Chavan, S. Huang, A. Deshpande, and A. Parameswaran. (2015). "Principles of dataset versioning: Exploring the recreation/storage tradeoff". *Proceedings of the VLDB Endowment*. 8(12): 1346–1357.

Bidoit, N., M. Herschel, and A. Tzompanaki. (2015). "Efficient Computation of Polynomial Explanations of Why-Not Questions". In: *Proceedings of the 24th ACM International Conference on Information and Knowledge Management, CIKM 2015, Melbourne, VIC, Australia, October 19 - 23, 2015*. Ed. by J. Bailey, A. Moffat, C. C. Aggarwal, M. de Rijke, R. Kumar, V. Murdock, T. K. Sellis, and J. X. Yu. ACM. 713–722. ISBN: 978-1-4503-3794-6. DOI: 10.1145/2806416.2806426. URL: https://doi.org/10.1145/2806416.2806426.

Bidoit, N., M. Herschel, K. Tzompanaki, *et al.* (2014). "Query-Based Why-Not Provenance with NedExplain". In: *Extending Database Technology (EDBT)*.

Bidoit, N., M. Herschel, and K. Tzompanaki. (2016). "Refining SQL Queries based on Why-Not Polynomials". In: *8th USENIX Workshop on the Theory and Practice of Provenance (TaPP 16)*. Washington, D.C.: USENIX Association.

Biton, O., S. Cohen-Boulakia, S. Davidson, and C. Hara. (2008). "Querying and Managing Provenance through User Views in Scientific Workflows". *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*: 1072–1081.

Biton, O., S. Cohen-Boulakia, and S. B. Davidson. (2007). "Zoom* userviews: Querying relevant provenance in workflow systems". In: *Proceedings of the 33rd international conference on Very large data bases*. VLDB Endowment. 1366–1369.

Borkin, M. A., C. S. Yeh, M. Boyd, P. Macko, K. Z. Gajos, M. I. Seltzer, and H. Pfister. (2013). "Evaluation of Filesystem Provenance Visualization Tools". *IEEE Trans. Vis. Comput. Graph.* 19(12): 2476–2485. DOI: 10.1109/TVCG.2013.155. URL: https://doi.org/10.1109/TVCG.2013.155.

Bourhis, P., D. Deutch, and Y. Moskovitch. (2020). "Equivalence-Invariant Algebraic Provenance for Hyperplane Update Queries". In: *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference Portland, OR, USA, June 14-19, 2020*. Ed. by D. Maier, R. Pottinger, A. Doan, W.-C. Tan, A. Alawini, and H. Q. Ngo. ACM. 415–429. ISBN: 978-1-4503-6735-6. DOI: 10.1145/3318464.3380578. URL: https://doi.org/10.1145/3318464.3380578.

Brachmann, M., C. Bautista, S. Castelo, S. Feng, J. Freire, B. Glavic, O. Kennedy, H. Müller, R. Rampin, W. Spoth, and Y. Yang. (2019). "Data Debugging and Exploration with Vizier". In: *Proceedings of the 44th International Conference on Management of Data (Demonstration Track)*.

Brain, M., M. Gebser, J. Pührer, T. Schaub, H. Tompits, and S. Woltran. (2007). "Debugging ASP Programs by Means of ASP". In: *Logic Programming and Nonmonotonic Reasoning, 9th International Conference, LPNMR 2007, Tempe, AZ, USA, May 15-17, 2007, Proceedings*. Ed. by C. Baral, G. Brewka, and J. S. Schlipf. Vol. 4483. *Lecture Notes in Computer Science*. Springer. 31–43. ISBN: 978-3-540-72199-4. DOI: 10.1007/978-3-540-72200-7_5. URL: https://doi.org/10.1007/978-3-540-72200-7%5C_5.

Brain, M. and M. D. Vos. (2005). "Debugging Logic Programs under the Answer Set Semantics". In: *Answer Set Programming, Advances in Theory and Implementation, Proceedings of the 3rd Intl. ASP'05 Workshop, Bath, UK, September 27-29, 2005*. Ed. by M. D. Vos and A. Provetti. Vol. 142. *CEUR Workshop Proceedings*. CEUR-WS.org. URL: http://ceur-ws.org/Vol-142/page141.pdf.

Buneman, P., J. Cheney, and S. Vansummeren. (2008). "On the Expressiveness of Implicit Provenance in Query and Update Languages". *ACM Transactions on Database Systems (TODS)*. 33(4): 1–47.

Buneman, P., S. Khanna, and W.-C. Tan. (2001). "Why and Where: A Characterization of Data Provenance". In: *ICDT*. 316–330.

Buneman, P., S. Khanna, and W.-C. Tan. (2002). "On Propagation of Deletions and Annotations through Views". In: *PODS '02: Proceedings of the 21th Symposium on Principles of Database Systems*. 150–158.

Buneman, P., E. V. Kostylev, and S. Vansummeren. (2013). "Annotations are relative". In: *Proceedings of the 16th International Conference on Database Theory*. ACM. 177–188.

Caballero, R., Y. Garcia-Ruiz, and F. Saenz-Perez. (2015). "Debugging of Wrong and Missing Answers for Datalog Programs with Constraint Handling Rules". In: *Proceedings of the 17th International Symposium on Principles and Practice of Declarative Programming. PPDP '15*. Siena, Italy: ACM. 55–66. ISBN: 978-1-4503-3516-4.

Callahan, S. P., J. Freire, E. Santos, C. E. Scheidegger, C. T. Silva, and H. T. Vo. (2006). "Managing the evolution of dataflows with vistrails". In: *Data Engineering Workshops, 2006. Proceedings. 22nd International Conference on*. IEEE. 71–71.

Cate, B. ten, C. Civili, E. Sherkhonov, and W.-C. Tan. (2015). "High-level why-not explanations using ontologies". In: *Proceedings of the 34th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*. ACM. 31–43.

Ceri, S., G. Gottlob, and L. Tanca. (1989). "What you always wanted to know about Datalog(and never dared to ask)". *IEEE Transactions on Knowledge and Data Engineering*. 1(1): 146–166.

Chandra, A. K. and D. Harel. (1985). "Horn Clauses Queries and Generalizations". *J. Log. Program.* 2(1): 1–15. DOI: 10.1016/0743-1066(85)90002-0. URL: https://doi.org/10.1016/0743-1066(85)90002-0.

Chapman, A. and H. V. Jagadish. (2009). "Why Not?" In: *SIGMOD '09: Proceedings of the 35th SIGMOD International Conference on Management of Data*. 523–534.

Chapman, A., H. V. Jagadish, and P. Ramanan. (2008). "Efficient Provenance Storage". In: *SIGMOD '08: Proceedings of the 35th SIGMOD International Conference on Management of Data*. 993–1006.

Chavan, A., S. Huang, A. Deshpande, A. J. Elmore, S. Madden, and A. G. Parameswaran. (2015). "Towards a Unified Query Language for Provenance and Versioning". In: *7th USENIX Workshop on the Theory and Practice of Provenance, TaPP 2015, Edinburgh, Scotland, UK, July 8-9, 2015*. Ed. by P. Missier and J. Zhao. USENIX Association. URL: https://www.usenix.org/conference/tapp15/workshop-program/presentation/chavan.

Cheney, J. (2007). "Program Slicing and Data Provenance". *IEEE Data Engineering Bulletin*. 30(4): 22–28.

Cheney, J., A. Ahmed, and U. A. Acar. (2014). "Database Queries that Explain their Work". In: *Proceedings of the 16th International Symposium on Principles and Practice of Declarative Programming, Kent, Canterbury, United Kingdom, September 8-10, 2014*. Ed. by O. Chitil, A. King, and O. Danvy. ACM. 271–282. ISBN: 978-1-4503-2947-7. DOI: 10.1145/2643135.2643143. URL: https://doi.org/10.1145/2643135.2643143.

Cheney, J., L. Chiticariu, and W.-C. Tan. (2009). "Provenance in Databases: Why, How, and Where". *Foundations and Trends in Databases*. 1(4): 379–474.

Chirigati, F. S., D. Shasha, and J. Freire. (2013). "ReproZip: Using Provenance to Support Computational Reproducibility." In: *TaPP*.

Chiticariu, L. and W.-C. Tan. (2006). "Debugging Schema Mappings with Routes". In: *VLDB '06: Proceedings of the 32th International Conference on Very Large Data Bases*. 79–90.

Chiticariu, L., W.-C. Tan, and G. Vijayvargiya. (2005). "DBNotes: a Post-it System for Relational Databases based on Provenance". In: *SIGMOD '05: Proceedings of the 31th SIGMOD International Conference on Management of Data*. 942–944.

Chockler, H. and J. Halpern. (2004). "Responsibility and blame: A structural-model approach". *Journal of Artificial Intelligence Research*. 22(1): 93–115. ISSN: 1076-9757.

Chockler, H., J. Halpern, and O. Kupferman. (2008). "What causes a system to satisfy a specification?" *ACM Transactions on Computational Logic (TOCL)*. 9(3): 1–26. ISSN: 1529-3785.

Chomsky, N. and M. P. Schützenberger. (1959). "The algebraic theory of context-free languages". In: *Studies in Logic and the Foundations of Mathematics*. Vol. 26. Elsevier. 118–161.

Chu, S., B. Murphy, J. Roesch, A. Cheung, and D. Suciu. (2018). "Axiomatic foundations and algorithms for deciding semantic equivalences of SQL queries". *Proceedings of the VLDB Endowment*. 11(11): 1482–1495.

Comini, M., G. Levi, and G. Vitiello. (1995). "Efficient Detection of Incompleteness Errors in the Abstract Debugging of Logic Programs". In: *Proceedings of the Second International Workshop on Automated Debugging, AADEBUG 1995, Saint Malo, France, May 22-24, 1995*. 159–174.

Cong, G., W. Fan, F. Geerts, J. Li, and J. Luo. (2012). "On the Complexity of Annotation Propagation and View Update Analyses". *IEEE Transactions on Knowledge and Data Engineering*.

Cui, Y. and J. Widom. (2000a). "Practical Lineage Tracing in Data Warehouses". In: *ICDE*. 367–378.

Cui, Y. and J. Widom. (2000b). "Storing Auxiliary Data for Efficient Maintenance and Lineage Tracing of Complex Views". In: *DMDW '00: Proceedings of the 2th International Workshop on Design and Management of Data Warehouses*.

Cui, Y. and J. Widom. (2001). "Run-time Translation of View Tuple Deletions using Data Lineage". *Tech. rep.* Stanford University.

Cui, Y., J. Widom, and J. L. Wiener. (2000). "Tracing the Lineage of View Data in a Warehousing Environment". *TODS*. 25(2): 179–227.

Damásio, C. V., A. Analyti, and G. Antoniou. (2013). "Justifications for logic programming". In: *Logic Programming and Nonmonotonic Reasoning*. Springer. 530–542.

Davison, A. P., M. Mattioni, D. Samarkanov, and B. Telenczuk. (2014). "Sumatra: A Toolkit for Reproducible Research". *Implementing Reproducible Research*: 57.

Dean, J. and S. Ghemawat. (2004). "MapReduce: simplified data processing on large clusters". In: *Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation - Volume 6. OSDI'04*. San Francisco, CA.

Deutch, D., Z. Ives, T. Milo, and V. Tannen. (2013a). "Caravan: Provisioning for What-If Analysis". *CIDR '13*.

Deutch, D. and N. Frost. (2019). "Constraints-based explanations of classifications". In: *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE. 530–541.

Deutch, D., N. Frost, and A. Gilad. (2017). "Provenance for Natural Language Queries". *Proceedings of the VLDB Endowment*. 10(5).

Deutch, D., N. Frost, and A. Gilad. (2020). "Explaining Natural Language Query Results". *VLDB J.* 29(1): 485–508. DOI: 10.1007/s00778-019-00584-7. URL: https://doi.org/10.1007/s00778-019-00584-7.

Deutch, D., N. Frost, A. Gilad, and T. Haimovich. (2018a). "Nlprovenans: Natural Language Provenance for Non-Answers". *Proc. VLDB Endow.* 11(12): 1986–1989. DOI: 10.14778/3229863.3236241. URL: https://doi.org/10.14778/3229863.3236241.

Deutch, D. and A. Gilad. (2016). "QPlain: Query by explanation". In: *32nd IEEE International Conference on Data Engineering, ICDE 2016, Helsinki, Finland, May 16-20, 2016*. 1358–1361. DOI: 10.1109/ICDE.2016.7498344. URL: https://doi.org/10.1109/ICDE.2016.7498344.

Deutch, D. and A. Gilad. (2019). "Reverse-Engineering Conjunctive Queries from Provenance Examples". In: *Advances in Database Technology - 22nd International Conference on Extending Database Technology, EDBT 2019, Lisbon, Portugal, March 26-29, 2019*. 277–288. DOI: 10.5441/002/edbt.2019.25. URL: https://doi.org/10.5441/002/edbt.2019.25.

Deutch, D., A. Gilad, and Y. Moskovitch. (2015a). "Selective Provenance for Datalog Programs Using Top-K Queries". *Proceedings of the VLDB Endowment*. 8(12).

Deutch, D., A. Gilad, and Y. Moskovitch. (2015b). "selP: Selective tracking and presentation of data provenance". In: *Data Engineering (ICDE), 2015 IEEE 31st International Conference on*. IEEE. 1484–1487.

Deutch, D., A. Gilad, and Y. Moskovitch. (2018b). "Efficient Provenance Tracking for Datalog Using Top-K Queries". *VLDB J.* 27(2): 245–269. DOI: 10.1007/s00778-018-0496-7. URL: https://doi.org/10.1007/s00778-018-0496-7.

Deutch, D., T. Milo, S. Roy, and V. Tannen. (2014). "Circuits for Datalog Provenance". In: *ICDT*. 201–212.

Deutch, D., Y. Moskovitch, and V. Tannen. (2013b). "PROPOLIS: Provisioned Analysis of Data-Centric Processes". *Proceedings of the VLDB Endowment*. 6(12).

Diestelkämper, R. and M. Herschel. (2020). "Tracing nested data with structural provenance for big data analytics". In: *Proceedings of the 23rd International Conference on Extending Database Technology, EDBT 2020, Copenhagen, Denmark, March 30 - April 02, 2020*. Ed. by A. Bonifati, Y. Zhou, M. A. V. Salles, A. Böhm, D. Olteanu, G. H. L. Fletcher, A. Khan, and B. Yang. OpenProceedings.org. 253–264. ISBN: 978-3-89318-083-7. DOI: 10.5441/002/edbt.2020.23. URL: https://doi.org/10.5441/002/edbt.2020.23.

Dietrich, B. and T. Grust. (2015). "A SQL Debugger Built from Spare Parts: Turning a SQL: 1999 Database System into Its Own Debugger". In: *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. ACM. 865–870.

Dignös, A., B. Glavic, X. Niu, M. H. Böhlen, and J. Gamper. (2019). "Snapshot Semantics for Temporal Multiset Relations". *Proceedings of the VLDB Endowment*. 12(6): 639–652.

Dong, X. L. and D. Srivastava. (2013). "Compact explanation of data fusion decisions". In: *Proceedings of the 22nd international conference on World Wide Web*. International World Wide Web Conferences Steering Committee. 379–390.

Drabent, W., S. Nadjm-Tehrani, and J. Maluszynski. (1988). "Algorithmic Debugging with Assertions". In: *Meta-Programming in Logic Programming, Workshop on Meta-Programming in Logic, META 1988, University of Bristol, 22-24 June, 1988*. 501–521.

El Gebaly, K., P. Agrawal, L. Golab, F. Korn, and D. Srivastava. (2014). "Interpretable and informative explanations of outcomes". *Proceedings of the VLDB Endowment*. 8(1).

El Gebaly, K., G. Feng, L. Golab, F. Korn, and D. Srivastava. (2018). "Explanation Tables". *Sat*. 5: 14.

Eltabakh, M. Y., M. Ouzzani, W. G. Aref, A. K. Elmagarmid, Y. Laura-Silva, M. U. Arshad, D. Salt, and I. Baxter. (2008). "Managing Biological Data using BDBMS". In: *ICDE '08: Proceedings of the 24th International Conference on Data Engineering (demonstration)*. 1600–1603.

Esparza, J., S. Kiefer, and M. Luttenberger. (2007). "On fixed point equations over commutative semirings". In: *STACS 2007*. 296–307.

Esparza, J., M. Luttenberger, and M. Schlund. (2014). "FPsolve: A Generic Solver for Fixpoint Equations over Semirings". In: *Implementation and Application of Automata*. 1–15.

Fabbri, D. and K. LeFevre. (2011). "Explanation-based auditing". *Proceedings of the VLDB Endowment*. 5(1): 1–12.

Fagin, R., P. Kolaitis, L. Popa, and W. Tan. (2005a). "Composing schema mappings: Second-order dependencies to the rescue". *ACM Transactions on Database Systems (TODS)*. 30(4): 994–1055.

Fagin, R., P. G. Kolaitis, R. J. Miller, and L. Popa. (2005b). "Data Exchange: Semantics and Query Answering". *Theoretical Computer Science*. 336(1): 89–124.

Farnadi, G., B. Babaki, and L. Getoor. (2018). "Fairness in Relational Domains". In: *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society, AIES 2018, New Orleans, LA, USA, February 02-03, 2018*. 108–114. DOI: 10.1145/3278721.3278733. URL: https://doi.org/10.1145/3278721.3278733.

Feng, S., A. Huber, B. Glavic, and O. Kennedy. (2019). "Uncertainty Annotated Databases - A Lightweight Approach for Approximating Certain Answers". In: *Proceedings of the 44th International Conference on Management of Data*.

Fernandez, C., F. J. Provost, and X. Han. (2019). "Counterfactual Explanations for Data-Driven Decisions". In: *Proceedings of the 40th International Conference on Information Systems, ICIS 2019, Munich, Germany, December 15-18, 2019*. URL: https://aisel.aisnet.org/icis2019/data%5C_science/data%5C_science/8.

Fink, R., L. Han, and D. Olteanu. (2012). "Aggregation in probabilistic databases via knowledge compilation". *Proceedings of the VLDB Endowment*. 5(5): 490–501.

Flum, J., M. Kubierschky, and B. Ludäscher. (1997). "Total and partial well-founded datalog coincide". In: *Database Theory—ICDT'97*. Springer. 113–124.

Foster, J. N., T. J. Green, and V. Tannen. (2008). "Annotated XML: Queries and Provenance". In: *PODS '08: Proceedings of the 27th Symposium on Principles of Database Systems*. 271–280.

Freire, C., W. Gatterbauer, N. Immerman, and A. Meliou. (2015). "The complexity of resilience and responsibility for self-join-free conjunctive queries". *Proceedings of the VLDB Endowment*. 9(3): 180–191.

Freire, C., W. Gatterbauer, N. Immerman, and A. Meliou. (2020). "New Results for the Complexity of Resilience for Binary Conjunctive Queries with Self-Joins". In: *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2020, Portland, OR, USA, June 14-19, 2020*. 271–284. DOI: 10.1145/3375395.3387647. URL: https://doi.org/10.1145/3375395.3387647.

Freire, J., P. Bonnet, and D. Shasha. (2011). "Exploring the coming repositories of reproducible experiments: Challenges and opportunities". *Proc. VLDB Endow.* 4: 1494–1497.

Freire, J., P. Bonnet, and D. Shasha. (2012). "Computational reproducibility: state-of-the-art, challenges, and database research opportunities". In: *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*. ACM. 593–596.

Freire, J. and F. Chirigati. (2018). "Provenance and the Different Flavors of Computational Reproducibility". *Data Engineering*: 15.

Freire, J. and C. T. Silva. (2012). "Making Computations and Publications Reproducible with VisTrails". *Computing in Science and Engineering*. 14(4): 18–25.

Gawlick, D. and V. Radhakrishnan. (2011). "Fine Grain Provenance Using Temporal Databases". In: *TaPP '11: 3rd USENIX Workshop on the Theory and Practice of Provenance*.

Geerts, F., G. Karvounarakis, V. Christophides, and I. Fundulaki. (2012). "Algebraic Structures for Capturing the Provenance of SPARQL Queries". In: *Proceedings of the 16th International Conference on Database Theory*.

Geerts, F., A. Kementsietsidis, and D. Milano. (2006). "MONDRIAN: Annotating and querying databases through colors and blocks". In: *Data Engineering, 2006. ICDE'06. Proceedings of the 22nd International Conference on*. IEEE. 82–82.

Geerts, F. and A. Poggi. (2010). "On database query languages for K-relations". *Journal of Applied Logic*. 8(2): 173–185.

Glavic, B. (2010). "Perm: Efficient Provenance Support for Relational Databases". *PhD thesis*. University of Zurich.

Glavic, B. and G. Alonso. (2009a). "Perm: Processing Provenance and Data on the same Data Model through Query Rewriting". In: *Proceedings of the 25th IEEE International Conference on Data Engineering*. 174–185.

Glavic, B. and G. Alonso. (2009b). "Provenance for Nested Subqueries". In: *Proceedings of the 12th International Conference on Extending Database Technology*. 982–993.

Glavic, B. and G. Alonso. (2009c). "The Perm Provenance Management System in Action". In: *Proceedings of the 35th ACM SIGMOD International Conference on Management of Data (Demonstration Track)*. 1055–1058.

Glavic, B., G. Alonso, R. J. Miller, and L. M. Haas. (2010). "TRAMP: Understanding the Behavior of Schema Mappings through Provenance". *Proceedings of the Very Large Data Bases Endowment*. 3(1): 1314–1325.

Glavic, B., J. Du, R. J. Miller, G. Alonso, and L. M. Haas. (2011). "Debugging Data Exchange with Vagabond". *Proceedings of the VLDB Endowment (Demonstration Track)*. 4(12): 1383–1386.

Glavic, B., K. S. Esmaili, P. M. Fischer, and N. Tatbul. (2013a). "Ariadne: Managing Fine-Grained Provenance on Data Streams". In: *Proceedings of the 7th ACM International Conference on Distributed Event-Based Systems*. 291–320.

Glavic, B., S. Köhler, S. Riddle, and B. Ludäscher. (2015). "Towards Constraint-based Explanations for Answers and Non-Answers". In: *Proceedings of the 7th USENIX Workshop on the Theory and Practice of Provenance*.

Glavic, B., A. Meliou, and S. Roy. (2021). "Trends in Explanations". *Foundations and Trends® in Databases*: to appear.

Glavic, B., R. J. Miller, and G. Alonso. (2013b). "Using SQL for Efficient Generation and Querying of Provenance Information". *In search of elegance in the theory and practice of computation: a Festschrift in honour of Peter Buneman*: 291–320.

Gondran, M. and M. Minoux. (2008). *Graphs, dioids and semirings: new models and algorithms*. Vol. 41. Springer Science & Business Media.

Grädel, E. and V. Tannen. (2017). "Semiring Provenance for First-Order Model Checking". *arXiv preprint arXiv:1712.01980*.

Grädel, E. and V. Tannen. (2020). "Provenance analysis for logic and games". *Moscow Journal of Combinatorics and Number Theory*. 9(3): 203–228.

Green, T. (2011). "Containment of conjunctive queries on annotated relations". *Theory of Computing Systems*. 49(2): 429–459.

Green, T., G. Karvounarakis, and Z. Tannen. (2010). "Provenance in ORCHESTRA".

Green, T. J. and V. Tannen. (2017). "The Semiring Framework for Database Provenance". In: *PODS*. 93–99.

Green, T. J. (2009). "Containment of Conjunctive Queries on Annotated Relations". In: *ICDT '09: Proceedings of the 16th International Conference on Database Theory*. 296–309.

Green, T. J., Z. G. Ives, and V. Tannen. (2009). "Reconcilable Differences". In: *ICDT '09: Proceedings of the 16th International Conference on Database Theory*. Saint Petersburg, Russia. 212–224.

Green, T. J., G. Karvounarakis, and V. Tannen. (2007a). "Provenance Semirings". In: *PODS*. 31–40.

Green, T. J., G. Karvounarakis, N. E. Taylor, O. Biton, Z. G. Ives, and V. Tannen. (2007b). "ORCHESTRA: Facilitating Collaborative Data Sharing". In: *SIGMOD '07: Proceedings of the 33th SIGMOD International Conference on Management of Data*.

Grust, T., F. Kliebhan, J. Rittinger, and T. Schreiber. (2011). "True language-level SQL debugging". In: *Proceedings of the 14th International Conference on Extending Database Technology*. ACM. 562–565.

Gulzar, M. A., M. Interlandi, T. Condie, and M. Kim. (2017a). "Debugging Big Data Analytics in Spark with BigDebug". In: *Proceedings of the 2017 ACM International Conference on Management of Data*. ACM. 1627–1630.

Gulzar, M. A., M. Interlandi, X. Han, M. Li, T. Condie, and M. Kim. (2017b). "Automated debugging in data-intensive scalable computing". In: *Proceedings of the 2017 Symposium on Cloud Computing, SoCC 2017, Santa Clara, CA, USA, September 24-27, 2017*. ACM. 520–534. ISBN: 978-1-4503-5028-0. DOI: 10.1145/3127479.3131624. URL: https://doi.org/10.1145/3127479.3131624.

Guo, P. J., S. Kandel, J. M. Hellerstein, and J. Heer. (2011). "Proactive wrangling: Mixed-initiative end-user programming of data transformation scripts". In: *Proceedings of the 24th annual ACM symposium on User interface software and technology*. ACM. 65–74.

Gupta, A., I. S. Mumick, *et al.* (1995). "Maintenance of materialized views: Problems, techniques, and applications". *IEEE Data Eng. Bull.* 18(2): 3–18.

Halpern, J. (2000). "Axiomatizing causal reasoning". *Arxiv preprint cs/0005030*.

Halpern, J. and J. Pearl. (2005). "Causes and explanations: A structural-model approach. Part I: Causes". *The British journal for the philosophy of science*. 56(4): 843. ISSN: 0007-0882.

Heinis, T. and G. Alonso. (2008). "Efficient Lineage Tracking for Scientific Workflows". In: *SIGMOD '08: Proceedings of the 34th SIGMOD International Conference on Management of Data*. 1007–1018.

Hellerstein, J. M., V. Sreekanti, J. E. Gonzalez, J. Dalton, A. Dey, S. Nag, K. Ramachandran, S. Arora, A. Bhattacharyya, S. Das, M. Donsky, G. Fierro, C. She, C. Steinbach, V. Subramanian, and E. Sun. (2017). "Ground: A Data Context Service." In: *CIDR*.

Herschel, M. and M. Hernandez. (2010). "Explaining Missing Answers to SPJUA Queries". *PVLDB*. 3(1): 185–196.

Herschel, M. (2013). "Wondering why data are missing from query results?: ask conseil why-not". In: *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*. ACM. 2213–2218.

Herschel, M., R. Diestelkämper, and H. B. Lahmar. (2017). "A survey on provenance: What for? What form? What from?" *The VLDB Journal*: 1–26.

Herschel, M., M. A. Hernández, and W.-C. Tan. (2009). "Artemis: A System for Analyzing Missing Answers". In: *VLDB '09: Proceedings of the 35th International Conference on Very Large Data Bases (demonstration)*. 1550–1553.

Hoekstra, R. and P. Groth. (2014). "PROV-O-Viz - Understanding the Role of Activities in Provenance". In: *Provenance and Annotation of Data and Processes - 5th International Provenance and Annotation Workshop, IPAW 2014, Cologne, Germany, June 9-13, 2014. Revised Selected Papers*. Ed. by B. Ludäscher and B. Plale. Vol. 8628. *Lecture Notes in Computer Science*. Springer. 215–220. ISBN: 978-3-319-16461-8. DOI: 10.1007/978-3-319-16462-5_18. URL: https://doi.org/10.1007/978-3-319-16462-5%5C_18.

Huang, J., T. Chen, A. Doan, and J. F. Naughton. (2008). "On the Provenance of Non-answers to Queries over Extracted Data". *PVLDB: Proceedings of the VLDB Endowment archive*. 1(1): 736–747.

Hull, R. and M. Yoshikawa. (1990). "ILOG: Declarative Creation and Manipulation of Object Identifiers". In: *16th International Conference on Very Large Data Bases, August 13-16, 1990, Brisbane, Queensland, Australia, Proceedings*. Morgan Kaufmann. 455–468. ISBN: 1-55860-149-X. URL: http://www.vldb.org/conf/1990/P455.PDF.

Ibrahim, K., X. Du, and M. Eltabakh. (2015). "Proactive Annotation Management in Relational Databases". In: *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. ACM. 2017–2030.

Ikeda, R., H. Park, and J. Widom. (2011a). "Provenance for generalized map and reduce workflows". In: *CIDR*. 273–283.

Ikeda, R., S. Salihoglu, and J. Widom. (2011b). "Provenance-based refresh in data-oriented workflows". In: *Proceedings of the 20th ACM international conference on Information and knowledge management. CIKM '11*. Glasgow, Scotland, UK: ACM. 1659–1668. ISBN: 978-1-4503-0717-8.

Ikeda, R. and J. Widom. (2010). "Panda: A System for Provenance and Data". In: *TaPP '10*. Stanford InfoLab.

Imieliński, T. and W. Lipski Jr. (1984). "Incomplete Information in Relational Databases". *Journal of the ACM (JACM)*. 31(4): 761–791.

Interlandi, M., A. Ekmekji, K. Shah, M. A. Gulzar, S. D. Tetali, M. Kim, T. D. Millstein, and T. Condie. (2018). "Adding data provenance support to Apache Spark". *VLDB J.* 27(5): 595–615. DOI: 10.1007/s00778-017-0474-5. URL: https://doi.org/10.1007/s00778-017-0474-5.

Interlandi, M., K. Shah, S. D. Tetali, M. A. Gulzar, S. Yoo, M. Kim, T. Millstein, and T. Condie. (2016). "Titian: Data Provenance Support in Spark". *PVLDB*. 9(3).

Ives, Z., A. Haeberlen, T. Feng, and W. Gatterbauer. (2012). "Querying provenance for ranking and recommending".

Ives, Z. G., T. J. Green, G. Karvounarakis, N. E. Taylor, V. Tannen, P. P. Talukdar, M. Jacob, and F. Pereira. (2008). "The ORCHESTRA Collaborative Data Sharing System". *SIGMOD Record*. 37(2): 26–32.

Ives, Z. G., N. Khandelwal, A. Kapur, and M. Cakir. (2005). "ORCHESTRA: Rapid, Collaborative Sharing of Dynamic Data". In: *CIDR '05: Proceedings of the 2th Conference on Innovative Data Systems Research*.

Jagadish, H. V., F. Bonchi, T. Eliassi-Rad, L. Getoor, K. P. Gummadi, and J. Stoyanovich. (2019). "The Responsibility Challenge for Data". In: *Proceedings of the 2019 International Conference on Management of Data, SIGMOD Conference 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019*. Ed. by P. A. Boncz, S. Manegold, A. Ailamaki, A. Deshpande, and T. Kraska. ACM. 412–414. ISBN: 978-1-4503-5643-5. DOI: 10.1145/3299869.3314327. URL: https://doi.org/10.1145/3299869.3314327.

Janin, Y., C. Vincent, and R. Duraffort. (2014). "CARE, the comprehensive archiver for reproducible execution". In: *Proceedings of the 1st ACM SIGPLAN Workshop on Reproducible Research Methodologies and New Publication Models in Computer Engineering, TRUST 2014, Edinburgh, United Kingdom, June 9-11, 2014*. Ed. by G. Fursin, B. R. Childers, A. K. Jones, and D. Mossé. ACM. 1:1–1:7. ISBN: 978-1-4503-2951-4. DOI: 10.1145/2618137.2618138. URL: https://doi.org/10.1145/2618137.2618138.

Jensen, C. and R. Snodgrass. (1999). "Temporal Data Management". *IEEE Transactions on Knowledge and Data Engineering*. 11(1): 36–44.

Joglekar, M., H. Garcia-Molina, and A. G. Parameswaran. (2019). "Interactive Data Exploration With Smart Drill-Down". *IEEE Trans. Knowl. Data Eng.* 31(1): 46–60. DOI: 10.1109/TKDE.2017.2685998. URL: https://doi.org/10.1109/TKDE.2017.2685998.

Kandel, S., A. Paepcke, J. Hellerstein, and J. Heer. (2011). "Wrangler: Interactive visual specification of data transformation scripts". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM. 3363–3372.

Karabeg, D. and V. Vianu. (1991). "Simplification Rules and Complete Axiomatization for Relational Update Transactions". *ACM Trans. Database Syst.* 16(3): 439–475. DOI: 10.1145/111197.111208. URL: https://doi.org/10.1145/111197.111208.

Karvounarakis, G. (2009). "Provenance in collaborative data sharing". *PhD thesis*. University of Pennsylvania.

Karvounarakis, G. and T. Green. (2012). "Semiring-Annotated Data: Queries and Provenance". *SIGMOD Record*. 41(3): 5–14.

Karvounarakis, G., Z. Ives, and V. Tannen. (2010). "Querying data provenance". In: *Proceedings of the 2010 international conference on Management of data*. ACM. 951–962.

Karvounarakis, G., T. J. Green, Z. G. Ives, and V. Tannen. (2013). "Collaborative data sharing via update exchange and provenance". *ACM Transactions on Database Systems (TODS)*. 38(3): 19.

Kaushik, R., Y. Fu, and R. Ramamurthy. (2013). "On scaling up sensitive data auditing". In: *Proceedings of the 39th international conference on Very Large Data Bases*. VLDB Endowment. 313–324.

Kaushik, R. and R. Ramamurthy. (2011). "Efficient auditing for complex SQL queries". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2011, Athens, Greece, June 12-16, 2011*. 697–708. DOI: 10.1145/1989323.1989396. URL: https://doi.org/10.1145/1989323.1989396.

Kemp, D. B., D. Srivastava, and P. J. Stuckey. (1995). "Bottom-Up Evaluation and Query Optimization of Well-Founded Models". *Theor. Comput. Sci.* 146(1&2): 145–184. DOI: 10.1016/0304-3975(94)00153-A. URL: https://doi.org/10.1016/0304-3975(94)00153-A.

Koh, P. W. and P. Liang. (2017). "Understanding Black-box Predictions via Influence Functions". In: *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*. Ed. by D. Precup and Y. W. Teh. Vol. 70. *Proceedings of Machine Learning Research*. PMLR. 1885–1894. URL: http://proceedings.mlr.press/v70/koh17a.html.

Köhler, S., B. Ludäscher, and Y. Smaragdakis. (2012). "Declarative datalog debugging for mere mortals". *Datalog in Academia and Industry*: 111–122.

Köhler, S., B. Ludäscher, and D. Zinn. (2013). "First-Order Provenance Games". In: *In Search of Elegance in the Theory and Practice of Computation*. Springer. 382–399.

Kolaitis, P. G. and C. H. Papadimitriou. (1988). "Why Not Negation by Fixpoint?" In: *Proceedings of the Seventh ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, March 21-23, 1988, Austin, Texas, USA*. Ed. by C. Edmondson-Yurkanan and M. Yannakakis. ACM. 231–239. ISBN: 0-89791-263-2. DOI: 10.1145/308386.308446. URL: https://doi.org/10.1145/308386.308446.

Koop, D. (2016). "Versioning Version Trees: The Provenance of Actions that Affect Multiple Versions". In: *Provenance and Annotation of Data and Processes - 6th International Provenance and Annotation Workshop, IPAW 2016, McLean, VA, USA, June 7-8, 2016, Proceedings*. Ed. by M. Mattoso and B. Glavic. Vol. 9672. *Lecture Notes in Computer Science*. Springer. 109–121. ISBN: 978-3-319-40592-6. DOI: 10.1007/978-3-319-40593-3_9. URL: https://doi.org/10.1007/978-3-319-40593-3%5C_9.

Kostylev, E. V. and P. Buneman. (2012). "Combining dependent annotations for relational algebra". In: *Proceedings of the 15th International Conference on Database Theory*. ACM. 196–207.

Kostylev, E. V., J. L. Reutter, and A. Z. Salamon. (2013). "Classification of Annotation Semirings over Containment of Conjunctive Queries". *TODS*.

Kunde, M., H. Bergmeyer, and A. Schreiber. (2008). "Requirements for a Provenance Visualization Component". In: *Provenance and Annotation of Data and Processes, Second International Provenance and Annotation Workshop, IPAW 2008, Salt Lake City, UT, USA, June 17-18, 2008. Revised Selected Papers*. Ed. by J. Freire, D. Koop, and L. Moreau. Vol. 5272. *Lecture Notes in Computer Science*. Springer. 241–252. ISBN: 978-3-540-89964-8. DOI: 10.1007/978-3-540-89965-5_25. URL: https://doi.org/10.1007/978-3-540-89965-5%5C_25.

Lee, S., S. Köhler, B. Ludäscher, and B. Glavic. (2017). "A SQL-Middleware Unifying Why and Why-Not Provenance for First-Order Queries". In: *Proceedings of the 33rd IEEE International Conference on Data Engineering*. 485–496.

Lee, S., B. Ludäscher, and B. Glavic. (2018). "PUG: a framework and practical implementation for why and why-not provenance". *The VLDB Journal*. 28(1): 47–71. ISSN: 0949-877X. DOI: 10.1007/s00778-018-0518-5.

Lee, S., B. Ludäscher, and B. Glavic. (2020). "Approximate Summaries for Why and Why-not Provenance". *Proceedings of the VLDB Endowment*. 13(6): 912–924.

Logothetis, D., S. De, and K. Yocum. (2013). "Scalable lineage capture for debugging DISC analytics". In: *Proceedings of the 4th annual Symposium on Cloud Computing*. ACM. 17.

Luttenberger, M. and M. Schlund. (2013). "Convergence of Newton's Method over Commutative Semirings". In: *Language and Automata Theory and Applications - 7th International Conference, LATA 2013, Bilbao, Spain, April 2-5, 2013. Proceedings*. 407–418. DOI: 10.1007/978-3-642-37064-9_36. URL: https://doi.org/10.1007/978-3-642-37064-9%5C_36.

Luttenberger, M. and M. Schlund. (2014). "Regular Expressions for Provenance". In: *TaPP*.

Meliou, A., W. Gatterbauer, K. Moore, and D. Suciu. (2010). "The Complexity of Causality and Responsibility for Query Answers and non-Answers". *Proceedings of the VLDB Endowment*. 4(1): 34–45.

Meliou, A. and D. Suciu. (2012). "Tiresias: The database oracle for how-to queries". In: *Proceedings of the 2012 international conference on Management of Data*. ACM. 337–348.

Meliou, A., W. Gatterbauer, S. Nath, and D. Suciu. (2011). "Tracing data errors with view-conditioned causality". In: *SIGMOD Conference*. Ed. by T. K. Sellis, R. J. Miller, A. Kementsietsidis, and Y. Velegrakis. ACM. 505–516. ISBN: 978-1-4503-0661-4.

Mohri, M. (2002). "Semiring Frameworks and Algorithms for Shortest-Distance Problems". *J. Autom. Lang. Comb.* 7(3): 321–350.

Moreau, L., B. Clifford, J. Freire, J. Futrelle, Y. Gil, P. Groth, N. Kwasnikowskag, S. Miles, P. Missier, J. Myers, B. Plale, Y. Simmhan, E. Stephan, and J. V. den Bussche. (2011). "The open provenance model core specification (v1. 1)". *Future Generation Computer Systems*. 27(6): 743–756.

Moreau, L., J. Freire, J. Futrelle, R. E. McGrath, J. Myers, and P. Paulson. (2008). "The Open Provenance Model: An Overview". In: *IPAW '08: International Provenance and Annotation Workshop*. 323–326.

Moreau, L., J. Freire, J. Myers, J. Futrelle, and P. Paulson. (2007). "The Open Provenance Model".

Moreau, L. and P. Groth. (2013). "Provenance: An introduction to prov". *Synthesis Lectures on the Semantic Web: Theory and Technology*. 3(4): 1–129.

Moreau, L. and P. Missier. (2013a). *http://www.w3.org/TR/prov-overview/*.

Moreau, L. and P. Missier. (2013b). "Prov-dm: The prov data model". *http://www.w3.org/TR/2013/REC-prov-dm-20130430/*.

Moura, L. M. de and N. Bjørner. (2011). "Satisfiability Modulo Theories: Introduction and Applications". *Commun. ACM*. 54(9): 69–77. DOI: 10.1145/1995376.1995394. URL: https://doi.org/10.1145/1995376.1995394.

Müller, T., B. Dietrich, and T. Grust. (2018). "You say what, i hear where and why: (mis-)interpreting SQL to derive fine-grained provenance". *Proceedings of the VLDB Endowment*. 11(11): 1536–1549.

Müller, T. and T. Grust. (2015). "Provenance for SQL through Abstract Interpretation: Value-less, but Worthwhile". *Proceedings of the VLDB Endowment*. 8(12).

Nguyen, D., J. Park, and R. Sandhu. (2013). "A provenance-based access control model for dynamic separation of duties". In: *Privacy, Security and Trust (PST), 2013 Eleventh Annual International Conference on*. IEEE. 247–256.

Niu, F., C. Zhang, C. Ré, and J. W. Shavlik. (2012). "DeepDive: Web-scale Knowledge-base Construction using Statistical Learning and Inference". In: *Proceedings of the Second International Workshop on Searching and Integrating New Web Data Sources, Istanbul, Turkey, August 31, 2012*. 25–28. URL: http://ceur-ws.org/Vol-884/VLDS2012%5C_p25%5C_Niu.pdf.

Niu, X., B. Glavic, S. Lee, B. Arab, D. Gawlick, Z. H. Liu, V. Krishnaswamy, S. Feng, and X. Zou. (2017a). "Debugging Transactions and Tracking their Provenance with Reenactment". *Proceedings of the VLDB Endowment (Demonstration Track)*. 10(12): 1857–1860.

Niu, X., R. Kapoor, B. Glavic, D. Gawlick, Z. H. Liu, V. Krishnaswamy, and V. Radhakrishnan. (2017b). "Provenance-aware Query Optimization". In: *Proceedings of the 33rd IEEE International Conference on Data Engineering*. 473–484.

Niu, X., R. Kapoor, B. Glavic, D. Gawlick, Z. H. Liu, V. Krishnaswamy, and V. Radhakrishnan. (2018). "Heuristic and Cost-based Optimization for Diverse Provenance Tasks". *IEEE Transactions on Knowledge and Data Engineering*. DOI: 10.1109/TKDE.2018.2827074.

Olston, C. and B. Reed. (2011). "Inspector Gadget: A Framework for Custom Monitoring and Debugging of Distributed Dataflows". *PVLDB*. 4(12): 1237–1248. URL: http://www.vldb.org/pvldb/vol4/p1237-olston.pdf.

Olteanu, D. and M. Schleich. (2016). "Factorized Databases". *ACM SIGMOD Record*. 45(2): 5–16.

Olteanu, D. and J. Závodny. (2011). "On Factorisation of Provenance Polynomials". In: *TaPP '11: 3rd USENIX Workshop on the Theory and Practice of Provenance*.

Olteanu, D. and J. Závodný. (2015). "Size Bounds for Factorised Representations of Query Results". *ACM Transactions on Database Systems (TODS)*. 40(1): 2.

Park, J., D. Nguyen, and R. Sandhu. (2012). "A provenance-based access control model". In: *Privacy, Security and Trust (PST), 2012 Tenth Annual International Conference on*. IEEE. 137–144.

Pasquier, T. F. J., X. Han, M. Goldstein, T. Moyer, D. M. Eyers, M. I. Seltzer, and J. Bacon. (2017). "Practical whole-system provenance capture". In: *Proceedings of the 2017 Symposium on Cloud Computing, SoCC 2017, Santa Clara, CA, USA, September 24-27, 2017*. ACM. 405–418. ISBN: 978-1-4503-5028-0. DOI: 10.1145/3127479.3129249. URL: https://doi.org/10.1145/3127479.3129249.

Pearl, J. (2000). *Causality: models, reasoning, and inference*. Cambridge Univ Pr. ISBN: 0521773628.

Pereira, L. M. (1986). "Rational Debugging in Logic Programming". In: *Third International Conference on Logic Programming, Imperial College of Science and Technology, London, United Kingdom, July 14-18, 1986, Proceedings*. 203–210. DOI: 10.1007/3-540-16492-8_76. URL: https://doi.org/10.1007/3-540-16492-8%5C_76.

Perera, R., U. Acar, J. Cheney, and P. Levy. (2012). "Functional programs that explain their work". In: *Proceedings of the 17th ACM SIGPLAN international conference on Functional programming*. ACM. 365–376.

Pham, Q., T. Malik, and I. Foster. (2013). "Using provenance for repeatability". In: *Proceedings of the 5th USENIX conference on Theory and Practice of Provenance*. 2–2.

Pham, Q., T. Malik, B. Glavic, and I. Foster. (2015). "LDV: Lightweight Database Virtualization". In: *Proceedings of the 31st IEEE International Conference on Data Engineering*. 1179–1190.

Pimentel, J. F., L. Murta, V. Braganholo, and J. Freire. (2019). "A large-scale study about quality and reproducibility of jupyter notebooks". In: *Proceedings of the 16th International Conference on Mining Software Repositories, MSR 2019, 26-27 May 2019, Montreal, Canada.* Ed. by M. D. Storey, B. Adams, and S. Haiduc. IEEE. 507–517. ISBN: 978-1-7281-3412-3. DOI: 10.1109/MSR.2019.00077. URL: https://doi.org/10.1109/MSR.2019.00077.

Psallidas, F. and E. Wu. (2018a). "Smoke: Fine-Grained Lineage At Interactive Speed". *Proc. VLDB Endow.* 11(6): 719–732. DOI: 10.14778/3184470.3184475. URL: https://doi.org/10.14778/3184470.3184475.

Psallidas, F. and E. Wu. (2018b). "Smoke: Fine-grained lineage at interactive speed". *Proceedings of the VLDB Endowment.* 11(6): 719–732.

Ramusat, Y., S. Maniu, and P. Senellart. (2018). "Semiring Provenance over Graph Databases". In: *10th USENIX Workshop on the Theory and Practice of Provenance, TaPP 2018, London, UK, July 11-12, 2018.* Ed. by M. Herschel. USENIX Association. URL: https://www.usenix.org/conference/tapp2018/presentation/ramusat.

Rio, N. D. and P. P. da Silva. (2007). "Probe-It! Visualization Support for Provenance". In: *Advances in Visual Computing, Third International Symposium, ISVC 2007, Lake Tahoe, NV, USA, November 26-28, 2007, Proceedings, Part II.* Ed. by G. Bebis, R. D. Boyle, B. Parvin, D. Koracin, N. Paragios, T. F. Syeda-Mahmood, T. Ju, Z. Liu, S. Coquillart, C. Cruz-Neira, T. Müller, and T. Malzbender. Vol. 4842. *Lecture Notes in Computer Science.* Springer. 732–741. ISBN: 978-3-540-76855-5. DOI: 10.1007/978-3-540-76856-2_72. URL: https://doi.org/10.1007/978-3-540-76856-2%5C_72.

Roy, S. and D. Suciu. (2014). "A formal approach to finding explanations for database queries". In: *SIGMOD.*

Salimi, B., L. Bertossi, D. Suciu, and G. V. den Broeck. (2016). "Quantifying Causal Effects on Query Answering in Databases". In: *8th USENIX Workshop on the Theory and Practice of Provenance (TaPP 16).* Washington, D.C.: USENIX Association.

Salimi, B., J. Gehrke, and D. Suciu. (2018). "Bias in OLAP Queries: Detection, Explanation, and Removal". In: *Proceedings of the 2018 International Conference on Management of Data*. ACM. 1021–1035.

Scheidegger, C. E., H. Vo, D. Koop, J. Freire, and C. T. Silva. (2008). "Querying and Re-using Workflows with VisTrails". In: *SIGMOD '08: Proceedings of the 34th SIGMOD International Conference on Management of Data*. ACM. 1251–1254.

Schwartz, E. J., T. Avgerinos, and D. Brumley. (2010). "All You Ever Wanted to Know about Dynamic Taint Analysis and Forward Symbolic Execution (but Might Have Been Afraid to Ask)". In: *31st IEEE Symposium on Security and Privacy, S&P 2010, 16-19 May 2010, Berleley/Oakland, California, USA*. IEEE Computer Society. 317–331. ISBN: 978-0-7695-4035-1. DOI: 10.1109/SP.2010.26. URL: https://doi.org/10.1109/SP.2010.26.

Senellart, P. (2017). "Provenance and Probabilities in Relational Databases: From Theory to Practice". *SIGMOD record*.

Senellart, P., L. Jachiet, S. Maniu, and Y. Ramusat. (2018). "ProvSQL: provenance and probability management in postgreSQL". *Proceedings of the VLDB Endowment*. 11(12): 2034–2037.

Shu, H. (2000). "Using constraint satisfaction for view update". *Journal of Intelligent Information Systems*. 15(2): 147–173. ISSN: 0925-9902.

Stonebraker, M., J. Chen, N. Nathan, C. Paxson, and J. Wu. (1993). "Tioga: Providing Data Management Support for Scientific Visualization Applications". In: *VLDB*. 25–38.

Suciu, D. (2020). "Probabilistic Databases for All". In: *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2020, Portland, OR, USA, June 14-19, 2020*. Ed. by D. Suciu, Y. Tao, and Z. Wei. ACM. 19–31. ISBN: 978-1-4503-7108-7. DOI: 10.1145/3375395.3389129. URL: https://doi.org/10.1145/3375395.3389129.

Suciu, D., D. Olteanu, C. Ré, and C. Koch. (2011). "Probabilistic databases". *Synthesis Lectures on Data Management*. 3(2): 1–180.

Suriarachchi, I., Q. Zhou, and B. Plale. (2015). "Komadu: A Capture and Visualization System for Scientific Data Provenance". *Journal of Open Research Software*. 3(1).

Tannen, V. (2017). "Provenance analysis for FOL model checking". *ACM SIGLOG News*. 4(1): 24–36.

That, D. H. T., G. Fils, Z. Yuan, and T. Malik. (2017). "Sciunits: Reusable Research Objects". In: *13th IEEE International Conference on e-Science, e-Science 2017, Auckland, New Zealand, October 24-27, 2017*. IEEE Computer Society. 374–383. ISBN: 978-1-5386-2686-3. DOI: 10.1109/eScience.2017.51. URL: https://doi.org/10.1109/eScience.2017.51.

Tran, Q. T. and C.-Y. Chan. (2010). "How to ConQueR why-not questions". In: *SIGMOD '10: Proceedings of the 2010 international conference on Management of data*. Indianapolis, Indiana, USA: ACM. 15–26. ISBN: 978-1-4503-0032-2.

Van den Broeck, G. and D. Suciu. (2017). "Query Processing on Probabilistic Data: A Survey".

Van Gelder, A., K. A. Ross, and J. S. Schlipf. (1991). "The well-founded semantics for general logic programs". *Journal of the ACM (JACM)*. 38(3): 619–649.

Vansummeren, S. and J. Cheney. (2007). "Recording Provenance for SQL Queries and Updates". *IEEE Data Engineering Bulletin*. 30(4): 29–37.

Vollmer, M., L. Golab, K. Böhm, and D. Srivastava. (2019). "Informative Summarization of Numeric Data". In: *Proceedings of the 31st International Conference on Scientific and Statistical Database Management, SSDBM 2019, Santa Cruz, CA, USA, July 23-25, 2019*. Ed. by C. Maltzahn and T. Malik. ACM. 97–108. ISBN: 978-1-4503-6216-0. DOI: 10.1145/3335783.3335797. URL: https://doi.org/10.1145/3335783.3335797.

Wang, Q., T. Yu, N. Li, J. Lobo, E. Bertino, K. Irwin, and J. Byun. (2007). "On the Correctness Criteria of Fine-Grained Access Control in Relational Databases". In: *Proceedings of the 33rd International Conference on Very Large Data Bases, University of Vienna, Austria, September 23-27, 2007*. Ed. by C. Koch, J. Gehrke, M. N. Garofalakis, D. Srivastava, K. Aberer, A. Deshpande, D. Florescu, C. Y. Chan, V. Ganti, C. Kanne, W. Klas, and E. J. Neuhold. ACM. 555–566. ISBN: 978-1-59593-649-3. URL: http://www.vldb.org/conf/2007/papers/research/p555-wang.pdf.

Wang, X., X. L. Dong, and A. Meliou. (2015). "Data X-Ray: A Diagnostic Tool for Data Errors". In: Sigmod.

Weiser, M. (1981). "Program slicing". *Proceedings of the 5th international conference on Software engineering*: 439–449.

Wu, E., S. Madden, and M. Stonebraker. (2012). "SubZero: A Fine-Grained Lineage System for Scientific Databases".

Wu, E. and S. Madden. (2013). "Scorpion: Explaining Away Outliers in Aggregate Queries". *PVLDB*. 6(8): 553–564.

Wu, Y., A. Haeberlen, W. Zhou, and B. T. Loo. (2013). "Answering why-not queries in software-defined networks with negative provenance". In: *Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks*. ACM. 3.

Wu, Y., M. Zhao, A. Haeberlen, W. Zhou, and B. T. Loo. (2014). "Diagnosing missing events in distributed systems with negative provenance". In: *Proceedings of the 2014 ACM conference on SIGCOMM*. ACM. 383–394.

Wu, Y., V. Tannen, and S. B. Davidson. (2020). "PrIU: A Provenance-Based Approach for Incrementally Updating Regression Models". In: *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*. Ed. by D. Maier, R. Pottinger, A. Doan, W. Tan, A. Alawini, and H. Q. Ngo. ACM. 447–462. ISBN: 978-1-4503-6735-6. DOI: 10.1145/3318464.3380571. URL: https://doi.org/10.1145/3318464.3380571.

Wylot, M., P. Cudré-Mauroux, and P. Groth. (2014). "TripleProv: Efficient Processing of Lineage Queries in a Native RDF Store". In: *Proceedings of the 24th international conference on World Wide Web (WWW)*.

Xu, J., W. Zhang, A. Alawini, and V. Tannen. (2018). "Provenance Analysis for Missing Answers and Integrity Repairs". *Data Engineering*: 39.

Yan, Z., V. Tannen, and Z. G. Ives. (2016). "Fine-grained Provenance for Linear Algebra Operators". In: *8th USENIX Workshop on the Theory and Practice of Provenance (TaPP 16)*. Washington, D.C.: USENIX Association.

Yang, Y., N. Meneghetti, R. Fehling, Z. H. Liu, and O. Kennedy. (2015). "Lenses: an on-demand approach to ETL". *Proceedings of the VLDB Endowment*. 8(12): 1578–1589.

Zaharia, M., M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica. (2010). "Spark: cluster computing with working sets". In: *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*. 10–10.

Zhang, J. and H. Jagadish. (2010). "Lost source provenance". In: *Proceedings of the 13th International Conference on Extending Database Technology*. ACM. 311–322.

Zhang, Z., E. R. Sparks, and M. J. Franklin. (2017). "Diagnosing machine learning pipelines with fine-grained lineage". In: *Proceedings of the 26th International Symposium on High-Performance Parallel and Distributed Computing*. ACM. 143–153.

Zheng, N., A. Alawini, and Z. G. Ives. (2019). "Fine-Grained Provenance for Matching & ETL". In: *35th IEEE International Conference on Data Engineering, ICDE 2019, Macao, China, April 8-11, 2019*. IEEE. 184–195. ISBN: 978-1-5386-7474-1. DOI: 10.1109/ICDE.2019.00025. URL: https://doi.org/10.1109/ICDE.2019.00025.

Index