Technical Report CS 01-22

Testing Provenance Systems

(Tanja Auge and Andreas Heuer)

Universität Rostock
Fakultät für Informatik und Elektrotechnik
Institut für Informatik
Lehrstuhl für Datenbank- und Informationssysteme

erhältlich unter:
https://eprints.dbis.informatik.uni-rostock.de/1076/

Testing Provenance Systems

Tanja Auge, Andraes Heuer

University of Rostock, Germany
{tanja.auge,andreas.heuer}@uni-rostock.de

Abstract. Data provenance allows to track the processing of individual data items. Meanwhile, there are more than 250 systems for evaluating provenance. In this report we study the most popular provenance systems including Tioga, Trio, Perm, GProM, Orchestra and ProvSQL for their functionality specified in the literature. Additionally, we test the different systems and classify or verify the provenance type of each one.

1 Introduction

Research institutions around the world produce research data in large quantities. Collecting, evaluating, analyzing, archiving and publishing these data are major tasks of research data management. The effort required to manage and store these data volumes is often underestimated, especially when certain criteria, such as the traceability or reproducibility of a published result, are to be guaranteed. This can be supported substantially by adding additional provenance.

Systems for determining ***why***-provenance do already exist. In this report we will give a short overview of the six best known provenance systems. For this, we will test them for their functionality and effectiveness. Most of them use witness basis [11] or polynomials [16] to introduce the provenance information.

This study supplements our publication *ProSA Pipeline: Provenance conquers the Chase* [6], were we introduce our own system ProSA. There we use additional provenance information for determining the minimal part of a database relevant for reconstructing a given query result. Using provenance makes it possible to reconstruct duplicates and remember specific attribute values necessary for inverting the query. Details about ProSA can be found in [5, 6, 8].

2 Data Provenance

"Provenance refers to all information describing the production process of a final product, which can be anything from digital data to physical objects" [17]. As summarized in [8], *data provenance* describes for a given database instance I and an evaluation query Q ***where*** a result tuple r does come from, ***why*** and ***how*** r exists in the result $Q(I)$.

A list of tuple identifiers such as $\{t_i, t_j, t_k\}$ is the result of ***where***-provenance. ***Why***-provenance [9] specifies a witness base such as $\{\{t_i, t_j\}, \{t_k\}\}$ that identifies the tuples involved in the calculation of r. The question of how a result tuple

r is calculated is answered by ***how***-provenance using provenance polynomials. These polynomials such as $t_i \cdot t_j + t_k$ give a concrete calculation of r. Here, they are defined by a commutative semi-ring $(\mathbb{N}[X], +, \cdot, 0, 1)$ with $+$ for union and projection as well as $\cdot$ for natural join [15].

The ***why***- and ***where***-provenance can be derived from the result of the ***how***-provenance. For this, we define a reduction based on the information content: ***where*** $\preccurlyeq$ ***why*** $\preccurlyeq$ ***how***. Therefore, the literature often focuses on ***why***- or ***how***-provenance. In the case of ***where***, we distinguish two types of provenance answers: The tuple-based ***where***-provenance stores the tuple identifiers. A reconstruction of concrete attribute values is not possible here. The same applies to the relation-based ***where***-provenance, which only returns the relation names.

3 Selecting the Systems

Since the field of provenance is still quite young, we will focus on the best known and most relevant provenance systems of the last 30 years. In 2017, the literature already refers to more than 250 provenance systems [22]. Based on this, we select six systems working on databases that are referenced particularly frequently, summarized in Fig. 1:

- Tioga resp. Tioga-2 is the oldest provenance system we could determine.
- Trio specializes in *data lineage*, a predecessor of today's ***why****-provenance*, as well as uncertainty.
- Orchestra is the first system that processes polynomials. It is thus a cornerstone of ***how****-provenance*.
- Perm uses annotation propagation and query rewriting to determine provenance and is also the predecessor of GProM, which implements an approach to generate provenance game graphs.
- ProvSQL is the latest provenance system, which was first introduced at ProvenanceWeek in 2018. It uses semi-rings to compute provenance polynomials, which are made particularly descriptive by user-defined aliases.

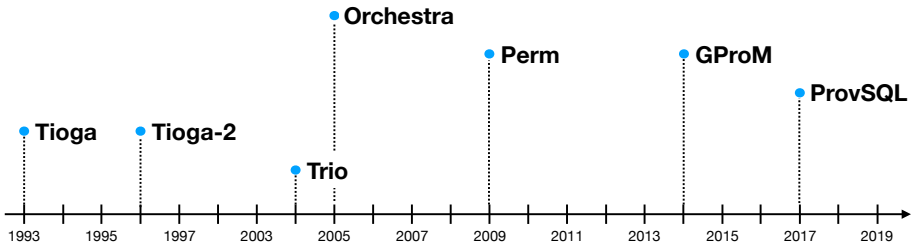


Fig. 1: Selection of tested provenance systems

4 Testing the Systems

Our tests aim to verify the functionality of the different systems specified in the literature. For this, we define four test queries: Query Q_1 tests duplicate elimination and query Q_2 processes a selection, a projection, and a natural join. By, joining three relations with projection on only the first and the third relation in query Q_3, we can show the differences of ***where***- and ***why***-provenance as well as data lineage. Finally, query Q_4 serves as a representative of simple aggregate functions such as `MAX`, `MIN`, `COUNT`, `SUM`, and `AVG`. The obtained results are summarized in Table 1. Thus, simple SPJ queries can be processed by all six systems. Unfortunately, because of graphical errors, a flawless execution of Tioga was not possible.

Table 1: Test query results

test query	Tioga	Trio	Orchestra	Perm	GProM	ProvSQL
Q_1	—	(✓)	✓	✓	✓	✓
Q_2	—	✓	✓	✓	✓	✓
Q_3	—	✓	✓	✓	✓	✓
Q_4	—	(✓)	✗	✓	✓	✗

For our concrete test scenario, let be GRADE(`studentID`, `courseID`, `grade`), STUDENT(`studentID`, `name`, `studies`), LECTURER(`courseID`, `lecturer`) as well as PARTICIPANT(`studentID`, `courseID`) be four relations of a fictional university database. Let further be I an arbitrary chosen instance with two courses 002 and 005 and at least two students named Jack.

Duplicate elimination (Q_1)

```
SELECT DISTINCT studies
FROM student;
```

Join, projection and selection (Q_2)

```
SELECT s.name
FROM student s JOIN participant p
ON s.studentID = p.studentID
WHERE p.courseID = '005';
```

Natural join of three relations (Q_3)

```
SELECT s.name, l.lecturer
FROM student s, participant p, lecturer l
WHERE s.studentID = p.studentID
AND p.courseID = l.courseID;
```

Aggregation and grouping (Q_4)

```
SELECT grade, count(*) AS gcount
FROM grade
WHERE courseID = '002'
GROUP BY grade
ORDER BY grade;
```

4.1 Tioga

The *Tioga system*[1], developed by Michael Stonebreaker et al., has two versions: Tioga and Tioga-2. The first one was introduced in 1993 as one of the first known provenance systems [23]. It was designed to "support data management of scientific visualizations" and was initially used to visualize the spread of forest fires, which is represented in the form of a directed graph. The representation reminds us of today's *workflow provenance* as defined in [17].

Tiogas redesign, called *Tioga-2* or *DataSplash* [2, 29], replaces the "box-arrow" notation with a direct manipulation programming paradigm, resulting in a more user-friendly, flexible, and powerful visualization system for manipulations. Furthermore, DataSplash is considered to be the first system representing the provenance of user-defined functions using inverse functions. According to [22], Tioga and its successors can also represent ***how***-provenance. However, we could not prove this in our investigations.

Applicability Since the project website of Tioga no longer exists and the source code was not available for us, we could not test Tioga ourselves. The installation of Tioga-2 worked very well, but it does not run without errors as seen in Fig. 2e.

4.2 Trio

Trio[2] was developed between 2004 and 2010 by Jennifer Widom et al. It extends PostgresSQL with *data lineage* as well as *uncertainty* and unified this in a graphical user interface called *TrioExplorer*. In addition, *TrioPlus* provides a command line interface, or you use short SQL scripts for interacting with Trio.

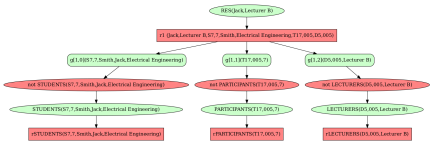
The data lineage implemented in Trio is stored in a separate table and can be displayed for each result tuples individually. As the definitions of ***why***- and ***how***-provenance as well as lineage have evolved over the years, the *Trio lineage* is not easy to define from today's perspective. It is more informative than the ***why***-, but less informative than the ***how***-provenance as seen in Fig. 2f.

The second special feature of Trio is its implementation of uncertainty. This allows the definition of alternative values for one, several, or all attributes of a database tuple as well as the specification of confidence values describing the

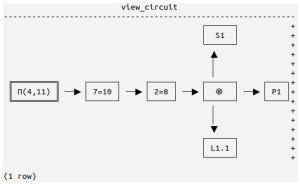
[1] Tioga: http://datasplash.cs.berkeley.edu
[2] Trio: http://infolab.stanford.edu/trio/

probability of the attribute's presence (highlighted green or orange in Fig. 2f). Both of these pieces of information will specify additional attributes.

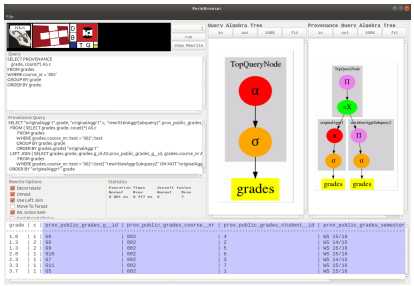
The *Trio Data Model* (TDM) is first introduced in [27] and then extended by a new data model called *Uncertainty Lineage Database Model (ULDB)* and a separate query language called *TriQL* [1, 10, 28]. Further developments of Trio are the two variants *Trio-ER* [7] and *LIVE* [26], which were developed from 2009. However, the best known and most used version is still the original.



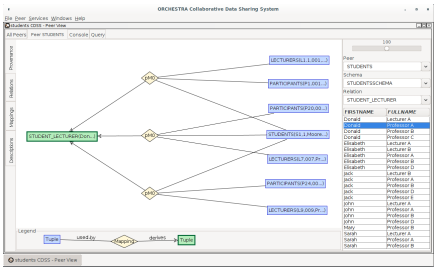
(a) Provenance game graph in GProM



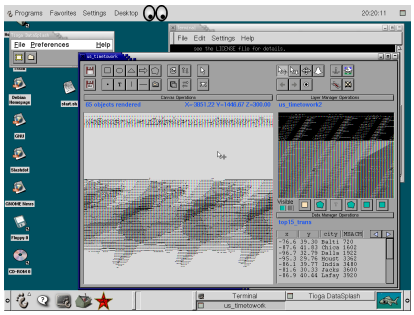
(b) Provenance graph in ProvSQL



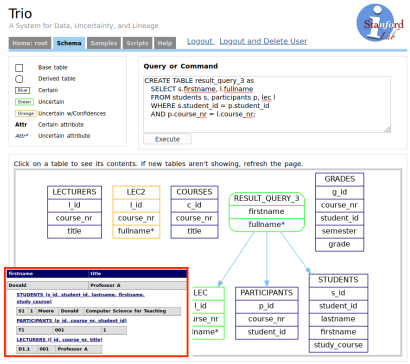
(c) Perm browser



(d) Provenance graph in Orchestra



(e) Incorrect visualization in DataSplash



(f) TrioExplorer

Fig. 2: Provenance representation in different systems

Applicability The Trio source code is open source and available on the Trio project page at Stanford University. When a query is executed, data lineage is automatically computed as well. An additional command is not necessary at this point. The output of one concrete result tuple can be seen in Fig. 2f (provenance schema on the right and provenance result of a concrete tuple on the left). In this case, the lineage indicates that the result tuple of query Q_3 consists of three source tuples of the relations STUDENTS, LECTURER and PARTICIPANT.

Trio processes simple SQL query — without allowing the `JOIN` keyword — including the aggregate functions `MIN`, `MAX`, `COUNT`, `SUM`, and `AVG`. However, the latter cannot be combined with uncertainty. Other than described in the documentation, we were not able to test the statements `DELETE` and `UPDATE`. Also the specification of foreign keys is not possible in our version of Trio.

All in all, Trio has a very clear user interface. Handling Trio is intuitive in terms of provenance investigation. Using uncertainty is a bit more complicated, but likewise straightforward.

4.3 Orchestra

To enable direct data exchange between two databases, *Orchestra*[3] tracks the exchanged provenance information. For this, it generates a provenance graph with tuple and mapping nodes as well as connecting edges. Internally, Orchestra is working with provenance semi-rings defined in [15]. It is thus the first system implementing the ***how***-provenance. The query language developed for navigating through the provenance graphs is called *ProQL*. It is presented in [19, 13, 14, 16] or [18]. Further, Orchestra eliminates duplicates and enables evaluating simple SPJU-queries. Aggregate functions, however, are not supported. The necessary polynomial theory was developed first in 2011 in [3].

Applicability We have tested Orchestra in version 0.1-SNAPSHOT. Since this is not the most recent version, some of our test results may differ from those in the literature. The latest version 0.2-SNAPSHOT does not work for us.

Orchestra creates a provenance graphs for each query result (highlighted green in Fig. 2d). The associated source tuples are highlighted in blue. Their connections or joins are marked by a yellow diamond. Duplicates are represented by arrows. For query Q_3, the result tuple exists three times, an information we get only indirectly in Trio.

The provenance graph corresponds to the provenance polynomial $L_1 \cdot P_1 \cdot S_1 + L_7 \cdot P_{20} \cdot S_1 + L_9 \cdot P_{24} \cdot S_1$. However, the representation in Fig. 2d differs from the one in the literature, because we were not able to reconstruct the operations $+$ and $\cdot$ instead of arrows and the diamonds. This might be due to the different versions of Orchestra.

[3] Orchestra: https://www.cis.upenn.edu/ zives/orchestra/

4.4 Perm

Implemented in 2009 as a modified PostgreSQL server, *Perm*[4] — **P**rovenance **E**xtension of the **R**elational **M**odel — is a provenance management system that supports provenance queries of various types. For this, a SQL query is transformed into a *SQL-PLE* query. The execution of this transformed query requires the additional keyword `PROVENANCE` in the `Select` clause. Thus, Perm uses the existing optimizer in PostgreSQL for query processing and additionally generates all necessary provenance information stored in additional provenance columns. The corresponding column names are composed of the prefix `prov`, the schema name, the source relation name and the attribute name such as *prov_public_grades_id*. So there is already some provenance information stored in the column name. Besides these additional columns, the Perm GUI also provides a query tree as well as a provenance query tree. In total, Perm implements the ***where***- and ***why***-provenance [12].

Applicability Perm describes how a query result is calculated. It handles simple SPJU queries as well as the classic aggregate functions `MAX`, `MIN`, `COUNT`, `SUM`, and `AVG`. Also sub-queries are possible. In addition, Perm has a user-friendly and clear GUI, seen in Fig. 2c. Due to the additional provenance columns duplicates cannot be detected directly. Accordingly, they are not eliminated and the query result given in Perm is by no means "minimal".

4.5 GProM

The *Generic Provenance Middleware* (GProM)[5] was developed as a successor to Perm. It's a middleware supporting various database management systems and languages such as SQL and Datalog, PostgresSQL, SQLite, Oracle, and MonetDB. GProM combines declarative queries with provenance queries and executes the resulting SQL code on a backend database system. According to the authors, GProM is the only tool computing provenance for SQL queries and transactions [4] by interpreting so-called *provenance games*. For this, a corresponding provenance graph [20] is created and colored depending on the provenance queries ***why*** or, ***why not***. Details can be found in [21].

Applicability The code of GProM is open source and available on GitHub. Due to its architecture, an additional front-end and a back-end are required. GProM itself is easy to install and well documented. Queries can be written in SQL as well as Datalog. An example of the provenance game graphs can be seen in Fig. 2a. It thus provides an implementation of ***why***- as well as ***why not***-provenance (depending on the coloring of the root node). We were able to reproduce all promised functions.

[4] Perm: https://github.com/IITDBGroup/perm
[5] GProM: https://github.com/IITDBGroup/gprom

4.6 ProvSQL

ProvSQL[6] is a PostgresSQL extension developed by Pierre Senellart et al. [25], which implements the theory of provenance semi-rings. The calculated polynomials are returned both as a formula and as a provenance graph, whereby each provenance ID can be replaced by a self-selected alias to increase it's readability.

Until today ProvSQL is continuously developed further. Currently it supports various semi-rings such as the *counting semiring*, the *tropical semiring* or *positive Boolean functions* (see [24, 16]). In addition, the specification of user-defined semiring [25] is possible. Further functionalities are expected.

Applicability PostgresSQL is provided as an additional feature. For this, an internal provenance ID is assigned with `add_provenance`. After that we define a mapping, which maps the provenance IDs to the user-selected alias using `create_provenance_mapping`. Finally, `formula()` computes the provenance polynomial we are looking for. An example graph can be seen in Fig. 2b. Summarized, ProvSQL processes simple SQL queries without any problems. Aggregate functions such as `MAX`, `AVG`, etc. cannot be processed yet (summer 2021), but are still being planned.

5 Summary

Meanwhile, there are more than 250 systems for evaluating provenance. We have tested the functionality of the six best-known systems: Tioga, Trio, Orchestra, Perm, GProM, and ProvSQL. As seen in Fig. 2, all systems except Tioga provide a graphical provenance representation. All systems can process simple SPJ queries such as Q_1, Q_2 and Q_3. The processing of queries with aggregate functions like Q_4, on the other hand, is only possible in the systems which are not based on provenance polynomials (see Table 1).

Table 2: Supported provenance types

Provenance	**Tioga**	**Trio**	**Orchestra**	**Perm**	**GProM**	**ProvSQL**
where	—	✓	✓	✓	✓	✓
why	—	✗	✗	✓	✓ (game graph)	✗
why not	—	✗	✗	✗	✓	✗
how	—	✗	✓ (graph)	✗	✗	✓ (formula & graph)
data lineage	—	✓	✗	✗	✗	✗

[6] ProvSQL: https://github.com/PierreSenellart/provsql/

Trio allows specifying data lineage, one of the oldest provenance types. Further, GProM provides a provenance game graph, which gives an answer to the question ***why*** or ***why not*** depending on the graph coloring. Perm prints all provenance tuples in additional provenance columns. This corresponds to a witness basis of ***why***-provenance. ProvSQL and Orchestra, on the other hand, answer the ***how*** a result ist calculated by specifying a provenance graph. In ProvSQL, the underlying provenance polynomial is defined as a formula represented by self-selected alias. The resulting classification of the different provenance types is summarized in Table 2.

Due to the provenance reduction of Section 2, the witness basis of the ***why***-provenance can always be derived from the provenance polynomial or provenance graphs of the ***how***-provenance. The same applies to the ***where***-provenance, which can be derived from the ***why***- or ***how***-provenance. ProvSQL and Orchestra, for example, satisfy the ***why***- and ***where***-provenance, even though this is not explicitly indicated by an ✓ in Table 2.

Acknowledgments We thank all students involved in testing the different provenance tools: Rocco Flach, Leon Herrmann, Chris Röhrs, Nic Scharlau, Artur Strelnikov.

References

1. Agrawal, P., Benjelloun, O., Das Sarma, A., Hayworth, Nabar, S. U., Sugihara, T., Widom, J.: Trio – A System for Data, Uncertainty, and Lineage, VLDB, pp. 1151–1154, ACM (2006)
2. Aiken, A., Chen, J., Stonebraker, M., Woodruff, A.: Tioga-2 – A Direct Manipulation Database Visualization Environment. In: ICDE, pp.208–217, IEEE Computer Society (1996)
3. Amsterdamer, Y., Deuch, D., Tannen, V.: Provenance for Aggregate Queries. In: PODS, pp. 153–164, ACM (2011)
4. Arab, B. S., Feng, S., Glavic, B., Lee, X., Niu, X., Zeng, Q.: GProM - A Swiss Army Knife for Your Provenance Needs. In: IEEE Data Eng. Bull. **41**(1), 51–62 (2018)
5. Auge, T., Heuer, A.: ProSA - Using the CHASE for Provenance Management. In: ADBIS, LNCS, vol. 11695, pp. 357–372. Springer (2019)
6. Auge, T., Hanzig, M., Heuer, A.: ProSA Pipeline – Provenance conquers the Chase. Accepted for ADBIS 2022
7. Agrawal, P., Ikeda, R., Park, H., Widom, J.: Trio-ER: The Trio System as a Workbench for Entity-Resolution. In: Technical Report, Stanford InfoLab, Stanford University, http://ilpubs.stanford.edu:8090/912/ (2009)
8. Auge, T.: Extended Provenance Management for Data Science Applications. In: PhD@VLDB, CEUR Workshop Proceedings, vol. 2652, CEUR-WS.org (2020)
9. Buneman, P., Khanna, S., Tan, W. C.: Why and Where – A Characterization of Data Provenance. In: ICDT, LNCS, vol. 1973, pp. 316–330, Springer (2001)
10. Benjelloun, O., Das Sarma, A., Hayworth, C., Widom, J.: An Introduction to ULDBs and the Trio System. In: IEEE Data Eng. Bull. **29**(1), 5–16 (2006)
11. Cheney, J., Chiticariu, L., Tan, W. C.: Provenance in Databases: Why, How, and Where. In: Found. Trends Databases **1**(4), 379–474 (2009)

12. Galvic, B., Alonso, G.: Perm – Processing Provenance and Data on the Same Data Model through Query Rewriting, ICDE, pp. 174–185, IEEE Computer Society (2009)
13. Green, T. J., Karvounarakis, G., Ives, Z. G., Tannen, V.: Update Exchange with Mappings and Provenance. In: VLDB, pp. 675–686, ACM (2007)
14. Green, T. J., Karvounarakis, G., Tannen, V.: Provenance semirings. In: PODS, pp. 31–40, ACM (2007)
15. Green, T. J., Karvounarakis, G., Tannen, V.: Provenance semirings. In: PODS, pp. 31–40, ACM (2007)
16. Green, T. J., Tannen, V.: The Semiring Framework for Database Provenance. In: PODS pp. 93–99, ACM (2017)
17. Herschel, M., Diestelkämper, R., Ben Lahmar, H.: A survey on provenance – What for? What form? What from?. In: VLDB J. **26**(6), 881–906 (2017)
18. Ives, Z. G., Green, T. J., Karvounarakis. G., Taylor, N. E., Tannen, V., Talukdar, P. P., Jacob, M., Pereira, F. C. N.: The ORCHESTRA Collaborative Data Sharing System. In: SIGMOD Rec. **37**(3), 26–32 (2008)
19. Ives, Z. G., Khandelwal, N., Kapur, A., Cakir, M.: ORCHESTRA: Rapid, Collaborative Sharing of Dynamic Data. In: CIDR, pp. 107–118, www.cidrdb.org (2005)
20. Köhler, S., Ludäscher, B., Zinn, D.: First-Order Provenance Games. In: In Search of Elegance in the Theory and Practice of Computation, LNCS, vol. 8000, pp. 382–399, Springer (2013)
21. Lee, S., Köhler, S., Ludäscher, B., Glavic, B.: A SQL-Middleware Unifying Why and Why-Not Provenance for First-Order Queries. In: ICDE, pp. 485–496 (2017)
22. Pérez, B., Rubio, J., Sáenz-Adán, C.: A systematic review of provenance systems. In: Knowl. Inf. Syst. **3**(57), 495–906 (2018)
23. Stonebraker, M., Chen, J., Nathan, N., Paxson, C., Wu, J.: Tioga – Providing Data Management Support for Scientific Visualization Applications. In: VLDB, pp.25-38, Morgan Kaufmann (1993)
24. Senellart, P.: Provenance and Probabilities in Relational Databases – From Theory to Practice. In: SIGMOD Record, vol. 4 (2017)
25. Senellart, P., Jachiet, L., Maniu, S., Ramusat, Y.: ProvSQL – Provenance and Probability Management in PostgreSQL. In: Proc. VLDB Endow. **11**(12), 2034–2037 (2018)
26. Das Sarma, A., Theobald, M., Widom, J.: LIVE – A Lineage-Supported Versioned DBMS. In: Technical Report, Stanford InfoLab, Stanford University, http://ilpubs.stanford.edu:8090/926/ (2009)
27. Widom, J.: Trio – A System for Integrated Management of Data, Accuracy, and Lineage. In: Technical Report, Stanford InfoLab, Stanford University, http://ilpubs.stanford.edu:8090/658/ (2004)
28. Widom, J.: Trio – A System for Data, Uncertainty, and Lineage. In: Managing and Mining Uncertain Data, Springer (2008)
29. Woodruff, A., Stonebraker, M.: Supporting Fine-grained Data Lineage in a Database Visualization Environment. In: ICDE, pp. 91–102, IEEE Computer Society(1997)