Traceable P2P Record Exchange Based on Database Technologies

Fengrong Li[1] and Yoshiharu Ishikawa[2]

[1] Graduate School of Information Science, Nagoya University
[2] Information Technology Center, Nagoya University
Furo-cho, Chikusa-ku, Nagoya 464-8601, Japan
lifr@db.itc.nagoya-u.ac.jp, ishikawa@itc.nagoya-u.ac.jp

Abstract. Information exchanges in P2P networks have become very popular in recent years. However, tracing how data circulates between peers and how data modifications are performed during the circulation before reaching the destination are not easy because data replications and modifications are performed independently by peers. This creates a lack of reliability among the records exchanged. To provide reliable and flexible information exchange facilities in P2P networks, we propose a framework for a record exchange system based on database technologies. The system consists of three layers: a user layer, a logical layer and a physical layer. Its tracing operations are executed as distributed recursive queries among cooperating peers in a P2P network. This paper describes the concept and overviews the framework.

1 Introduction

A *peer-to-peer* (*P2P*) network which consists of a large number of autonomous computers (*peers*) and is not dependent on a specific server is widely used in various applications such as file exchange, user communication, and content distribution. During information exchange in a P2P network, since duplications and changes to data may be performed by every peer without central control, it is difficult to determine the origin of data and to determine the movement of data between peers. This causes a lack of reliability in the data exchanged. As an example, when searching for images of beautiful scenes in a P2P file exchange service, reliability may not be significant, but when researchers exchange and share scientific information such as genome data with other researchers, the lack of reliability would be a critical concern. If a researcher is not sufficiently confident that the data was obtained from reliable sources, he will hesitate to use it for research purposes.

In this context, we propose a framework for *reliable record exchange* in P2P networks, where a *record* means a tuple-structured data item that obeys a predefined schema globally shared in the network. Records are exchanged between peers and peers can modify, store, and delete their records independently. The architecture of our P2P record exchange framework consists of three layers: the *user layer*, the *logical layer*, and the *physical layer*. The user layer provides a user interface for the record exchange system and the logical and physical layers support its internal representations. The two underlying layers are based on the relational data model, which is used for representing records in the user layer. In addition, they maintain record exchange and modification histories, facilitating traceability. In the physical layer, each peer in the P2P network maintains its own relations for storing information, and the logical layer provides virtual views by integrating the distributed relations. The abstraction in the logical layer provides a comprehensive framework for representing traceability requirements as database queries. Tracing queries are expressed as recursive *datalog* queries and executed over the distributed peers in the network.

The remainder of this paper is organized as follows. In Section 2, we describe the fundamental concept of P2P record exchange. In Section 3, we introduce the logical layer of the system. In Section 4, we describe the physical layer and present the concept underlying query processing. Section 5 reviews related work. Finally, in Section 6 we conclude the paper and outline future work.

2 P2P Record Exchange

2.1 Motivating Example

In this paper, we propose the concept of *traceable record exchange* in a P2P network and present the system architecture and its query processing framework. We assume that each peer corresponds to a user and maintains a set of *records* owned by the user. Each record has the same structure, which is defined by a predetermined schema which is globally shared within the network.

As an example, consider the sharing of information regarding novels among peers in a P2P network. Each peer maintains its own records and wishes to incorporate appropriate records from other peers to enhance its own record set. Figure 1 shows an example record set `Novel` owned by a peer that consists of four attributes: `title`, `author`, `language`, and `year`. Other peers also maintain their `Novel` records with the same structure, but their contents are not the same.

title	author	language	year
Pride and Prejudice	Jane Austen	English	1813
Madame Bovary	Gustave Flaubert	French	1857
War and Peace	Leo Tolstoy	Russian	1865

Fig. 1. Example Record Set `Novel`

In our record exchange framework, every peer can act as a provider of information. In this example, suppose that a user is interested in the novels written by `Jane Austen`. The user finds the desired records from other peers in the network by issuing a query. The user then examines the retrieved records which include `(Persuasion, Jane Austen, English, 1818)`, and the selected records are registered in the local record management system as additional records. Of course, the user can modify and/or delete the obtained records in the local record set. In addition, the user can allow access to the record set from the other peers in the network.

A *traceability problem* occurs, for example, when the owner of the record set shown in Fig. 1 has the question: "Was 'Pride and Prejudice' actually published in 1813?" The user might check that the record `(Pride and Prejudice, Jane Austen, English, 1813)` in Fig. 1 is correct and try to find evidence supporting its validity. If the record was obtained from a well-known and credible peer, the user may suppose that the record is probably correct. Alternatively, if the original creator of the record is reliable, the record would also be reliable. However, finding such evidence from a P2P network is quite difficult. Peers are highly distributed and there is no central server that can answer traceability queries using the complete histories of all the records in the network.

To solve the traceability problem, we propose a framework for record exchange with a traceability facility. All the information required for tracing is maintained in distributed peers. When a tracing query is given, it is executed with the cooperation of the peers in a distributed manner. The next subsection describes the overall framework of the record exchange system.

2.2 System Framework

The *record exchange system* consists of the following three layers:

1. *User layer*: offers a user interface for the record management system.
2. *Logical layer*: provides a virtual view containing whole records in the P2P network including information for tracing.
3. *Physical layer*: implements the logical virtual views based on the cooperation of autonomous peers.

We now briefly explain the function of the user layer. The user layer provides the following functions to the user:

– *Search*: The system executes given search queries using distributed peers. The details are beyond the scope of this paper.
– *Registration*: After registration, records are under the control of the system and are potential targets for tracing.
– *Deletion/Update*: A user can delete and modify the records in his local system.
– *Tracing*: The system provides tracing facilities to the user. The details are described below.

In the following discussion, we omit the details of the user layer since it is not the main topic of this paper. The two underlying layers, the logical layer and the physical layer, are described in detail in Sections 3 and 4.

2.3 Requirements for Traceability

If there is no support for traceability in record exchange, the following problems may occur:

1. *The source of a record cannot be determined*: For example, it is not clear whether a record which exists at peer A was created at peer A, or was obtained from other peers. Moreover, it is difficult to know which peer initially created the record. Since the record is insufficiently reliable, the advantages of record exchanging will be lost.
2. *Duplicate detection is not possible*: For example, when two or more records with the same values exist in a local record set, it is difficult to judge whether they were obtained from one single source or from different sources.
3. *The destination of a record cannot be identified*: Suppose that peer A discovers an error in one of its own records which is made available to other peers. If the peer wants to make error notifications to the peers that have incorporated the record, the identification of such peers is difficult.
4. *Updates cannot be traced*: Suppose that peer B has obtained a record from peer A and modified it, and that peer C has subsequently copied the modified record from peer B. Even if peer C wants to know the original source of the record, it is not possible because a simple search does not match the original value of the record at peer A.

The proposed record exchange framework copes with these problems using database technologies. The next section describes the logical layer, which represents information for tracing based on the relational model.

3 The Logical Layer

3.1 Data Representation

In the logical layer, virtual *views* are constructed by unifying the record sets (relations) maintained by distributed peers. By providing virtual integrated views, the users in the logical layer (e.g., the system administrator) can formulate queries for tracing more easily, as described in the following example. We simplify the example shown in Fig. 1 and assume that each peer maintains a `Novel` record set that has two attributes `title` and `author`. Figure 2 shows three record sets in the user layer maintained by peers A, B and C.

Peer A

title	author
t1	a1
t5	a5

Peer B

title	author
t1	a1
t2	a3

Peer C

title	author
t1	a1

Fig. 2. Record Sets among Three Peers

In the logical layer, records in the user layer are managed based on the relational data model. In each peer, three relational views are constructed and maintained. First, relation `Data[Novel]` in Fig. 3 expresses a view that unifies all the novel records held by peers A, B and C shown in Fig. 2. The view also contains old record values that were deleted or modified by users. They are hidden from the user layer but used for tracing histories. Attributes `title` and `author` are visible from the user layer, but the other two attributes `peer` and `id` are used for management. Attribute `peer` represents the logical name of the peer used for sending a message to other peers in the P2P network. Attribute `id` is used to identify records during query processing.

title	author	peer	id
t1	a1	A	#A011
t5	a5	A	#A028
t1	a1	B	#B032
t2	a2	B	#B040
t2	a3	B	#B051
t1	a1	C	#C005
t6	a6	C	#C077

Fig. 3. View `Data[Novel]`

from_id	to_id	time
−	#A011	5/2/07
−	#A028	8/18/07
−	#B032	4/10/07
−	#B040	4/20/07
#B040	#B051	6/10/07
−	#C005	3/20/07
#C077	−	10/06/07

Fig. 4. View `Change[Novel]`

Second, the relation `Change[Novel]` shown in Fig. 4 is a global view containing the insertion, modification, and deletion histories. Attributes `from_id` and `to_id` express the record ids before/after a modification. Attribute `time` represents the timestamp of the modification. When the value of the `from_id` attribute is the null value (−), it signifies that the record has been inserted. Similarly, when the value of the `to_id` attribute is the null value, it means that the record has been deleted.

Finally, view `Exchange[Novel]` shown in Fig. 5 stores information regarding record exchange among peers. Attributes `from_peer` and `to_peer` express the origin and the destination of record exchanges, respectively. Attributes `from_id` and `to_id` contain the logical ids of the exchanged record in both peers. Attribute `time` stores a timestamp expressing the time when the record was copied to the peer. For example, the first tuple shows that peer A copied the record from peer B, where it had the id value `#B032`, and peer A assigned a new id `#A011` for the record when it was registered at peer A. Record exchanges among peers can be traced using the three views.

from_peer	to_peer	from_id	to_id	time
B	A	#B032	#A011	5/2/07
C	B	#C005	#B032	4/10/07

Fig. 5. View `Exchange[Novel]`

3.2 Representation of Queries in the Logical Layer

In this subsection, we describe the representation of queries in the logical layer. Since recursive processing is needed in order to trace information in a network, queries are written using *datalog* [2, 9]. Datalog has been used for network-oriented query processing in fields such as in *declarative networking* [8]. We now present some example tracing queries.

Query 1: Suppose that peer A holds a record with title `t1` and author `a1` and that peer A wants to know which peer originally created the record. This query may be described as follows. Strings such as `P` and `I1` represent variables and '_' indicates an anonymous variable. The last rule represents the final result expected by the user.

```
BReach(P, I1) :- Data[Novel]('t1', 'a1', 'A', I2),
                 Exchange[Novel](P, 'A', I1, I2, _)
BReach(P1, I1) :- BReach(P2, I2), Exchange[Novel](P1, P2, I1, I2, _)
Origin(P) :- BReach(P, I), NOT Exchange[Novel](_, P, _, I)
Query(P) :- Origin(P)
```

Relation `BReach` defined by the first two rules means "Backward Reachable". In `BReach(P, I)`, the symbol `P` represents the name of the peer which was in the path from the originator of the record to peer A and `I` is the id of the record `(t1, a1)` when it was at peer `P`. In the first rule, the name of the peer which handed peer A the record directly is sought using the information in `Data[Novel]` and `Exchange[Novel]`. The second rule is for recursive processing. Thus, `BReach` collects information regarding all the peers which are in the path from the originator to peer A for the record in question. The third rule is used for selecting the peer at which the record originated. The originating peer should be reachable from peer A and should not have received the record from any other peer. If this query is processed according to the example views shown in Fig. 3, C is returned as the originating peer.

Query 2: Suppose that peer A wishes to know which of its own records were obtained via peer B. This query may be expressed as follows:

```
BReach2(P, I1, T, A) :- Data[Novel](T, A, 'A', I2),
                        Exchange[Novel](P, 'A', I1, I2, _)
BReach2(P1, I1, T, A) :- BReach2(P2, I2, T, A),
                         Exchange[Novel](P1, P2, I1, I2, _)
ViaB(T, A) :- BReach2('B', _, T, A)
Query(T, A) :- ViaB(T, A)
```

`BReach2` has a similar structure to `BReach` in Query 1. The difference is that `BReach2` holds additional information regarding novel titles and author names. `BReach2(P, I, T, A)` means that a record at peer A was a copy of the record with title `T` and author `A` held by peer `P` with id `I`. In the third rule, the titles and the authors which satisfy the constraints are extracted.

Query 3: In the process of P2P record exchange, there is a chance that a record `(t1, a1)` obtained from another peer in the past may be obtained again at some point. Suppose that peer A wishes to verify whether the recently obtained record is the same as the record already registered in its local system. It is easy to discover whether there is a record at peer A which has the same value with the given record `(t1, a1)` using a search query. However, even if the value is same, there is a possibility that some other peer has independently created a record with the same value. Suppose that peer A received a record `(t1, a1)` from peer D in which the record has the id `#D051`. The following query checks whether the originator of the given record `(t1, a1)` is the same as that of the record `(t1, a1)` at peer A with id `#A011`.

```
BReachA(P, I1) :- Data[Novel]('t1', 'a1', 'A', '#A011'),
                  Exchange[Novel](P, 'A', I1, I2, _)
BReachA(P1, I1) :- BReachA(P2, I2), Exchange[Novel](P1, P2, I1, I2, _)
BReachD(P, I1) :- Data[Novel]('t1', 'a1', 'D', '#D051'),
                  Exchange[Novel](P, 'D', I1, I2, _)
BReachD(P1, I1) :- BReachD(P2, I2), Exchange[Novel](P1, P2, I1, I2, _)
Dup :- BReachA(_, I), BReachD(_, I)
Query :- Dup
```

The first and the second rules collect the trace information regarding peer A. The third and the fourth rules play the same role for peer D. The fifth rule investigates whether peer A and peer D share a record id for identifying records. If this rule is satisfied, it can be concluded that the sources are the same.

Query 4: Suppose we wish to determine whether a record `(t1, a1)` held by peer A is the newest version. That is to say, we wish to determine whether some peer which gave the record to peer A, either directly or indirectly, has subsequently modified its record value. Suppose that peer A received the record `(t1, a1)` from some peer and that peer A wishes to determine whether the record was modified by any of the peers in the path from peer A to the originating peer.

The following query satisfies the requirement. `BReach3` is identical to `BReach`, shown above. The third rule detects if a record with id `I1` has been modified. The constraint that `I2` is not empty ensures that `I1` is not a deleted record.

```
BReach3(P,I1,T,A) :- Data[Novel]('t1', 'a1', 'A', I2),
                     Exchange[Novel](P, 'A', I1, I2, _),
                     Data[Novel](T, A, P, I1)
BReach3(P1,I1,T,A) :- BReach3(P2, I2,_,_),
                      Exchange[Novel](P1, P2, I1, I2, _),
                      Data[Novel](T, A, P1, I1)
Modified(P,T,A) :- BReach3(P, I1, T, A),
                   Change[Novel](P, I1, I2, _), I2 != NULL
Query(P,T,A) :- Modified(P,T,A)
```

4 The Physical Layer

4.1 Basic Idea

In the logical layer, queries are expressed using virtual views that unify all the information in a P2P network. However, it is inappropriate to materialize the views at a central server due to the following reasons.

– Each peer in a P2P network acts autonomously and functions cooperatively. A peer may not be interested in all the information in a network. Thus, it is not required to manage all the information in a central server.

– Although materialized views support efficient processing for tracing queries, such queries may be issued infrequently. Therefore, management of materialized views in this context may not be economical due to the high communication and processing cost.

In our framework, we assume that each peer in a network manages the information related to itself. Tracing queries in the logical layer are processed cooperatively by the peers in the physical layer.

4.2 Data Representation

The three relations in the logical layer are represented by four relations in the physical layer. The relations in Figs. 6 and 7 correspond to `Data[Novel]` and `Change[Novel]` in the logical layer, as shown in Figs. 3 and 4. These relations represent the information managed by peer A.

title	author	id
t1	a1	#A011
t5	a5	#A028

Fig. 6. `Data[Novel]` of Peer A

from_id	to_id	time
–	#A011	5/2/07
–	#A028	8/18/07

Fig. 7. `Change[Novel]` of Peer A

The content of `Exchange[Novel]` in the logical layer is partitioned and distributed among peers. The relation `From[Novel]` in the physical layer shown in Fig. 8 corresponds to `Exchange[Novel]` shown in Fig. 5 and contains records received by peer A. Figure 8 shows the record with id `#A011` is copied from the record with id `#B032` at peer B. In addition, each peer records information when it provides a record to another peer. Fig. 9 shows the `To[Novel]` relation of peer B, which corresponds to the `From[Novel]` relation of peer A shown in Fig. 8. It signifies the record with id `#B032` at peer B was copied by peer A with id `#A011`.

id	from_peer	from_id	time
#A011	B	#B032	5/2/07

Fig. 8. `From[Novel]` of Peer A

id	to_peer	to_id	time
#B032	A	#A011	5/2/07

Fig. 9. `To[Novel]` of Peer B

`From[Novel]` and `To[Novel]` contain duplicated information but are stored by different peers. In the above example, when peer A copies the record from peer B, the following steps are performed:

1. Peer A sends a query to peer B. Peer A obtains a record set from peer B.
2. Peer A selects the record with id `#B032` and registers it in its own local record management system.
3. Peer A assigns id `#A011` to the record and inserts the information into its `Data[Novel]` relation. The information that peer A received the record from peer B is then recorded in `From[Novel]` relation.
4. Peer A transmits the update information "`#B032` was registered with id `#A011` and timestamp `5/02/07`" to peer B.

5. Peer B records the information from peer A in its `To[Novel]` relation.

The reason for performing the process shown above is that peer B cannot know which records among those sent to peer A were actually registered at peer A, without the information provided in the message from peer A. The whole process is executed as a distributed transaction among peers A and B in order to ensure the consistency of the information stored.

4.3 Query Processing

In the logical layer, queries are described in datalog using virtual views that integrate all of the information in a P2P network. In order to process a tracing query, it is necessary to transform the given query to suit the organization of the physical layer. The approach for achieving this is described below.

Query Mapping to the Physical Layer Consider again Query 1 and assume that the query is issued at peer A. In order to map logical relations into physical relations, the query is translated into the following physical layer query:

```
BReach(P, I1) :- Data[Novel]@'A'('t1', 'a1', I2),
                 From[Novel]@'A'(I2, P, I1, _)
BReach(P1, I1) :- BReach(P2, I2), From[Novel]@P2(I2, P1, I1, _)
Origin(P) :- BReach(P, I), NOT From[Novel]@P(I, _, _, _)
Query(P) :- Origin(P)
```

Notation of the form `Data[Novel]@'A'` indicates a physical relation at a specific peer, in this case the relation `Data[Novel]` at peer A. Similarly, `From[Novel]@P` represents the relation `From[Novel]` at peer `P`. Note that `P` is a variable. The interpretation of the transformed query is straightforward: it traverses the path from peer A to the originator of the record `(t1, a1)` using recursive processing.

There are several options for the mapping. For example, we may replace every occurrence of `Exchange[Data]` in the logical query with `To[Novel]`. Although this produces another correct datalog query, it is hard to execute because it requires a traversal from the originator of the record to peer A.

Local Execution When seeking the local rule which can be processed at peer A, in this example, the first rule is a local rule and can be executed at peer A immediately. As a result, `BReach` will contain the information regarding which peers directly provided the record `(t1, a1)` to peer A. Assuming that the local evaluation result shown in Fig. 10 was obtained, we note that the contents of Fig. 10 do not show the correct result given the relations in Figs. 6 and 8, but are used below for ease of understanding.

Query Forwarding Peer A forwards the following sub-query corresponding to the transformed query to the peers that can directly execute it using the current contents of `BReach`.

P	I
B	#B032
C	#C128
B	#B093

Fig. 10. Relation `BReach` at Peer A

```
BReach(P1, I1) :- BReach(P2, I2), From[Novel]@P2(I2, P1, I1, _)
Origin(P) :- BReach(P, I), NOT From[Novel]@P(I, _, _, _)
```

In this case, peer B will receive the sub-query with a set $\{(\texttt{B}, \#\texttt{B032}), (\texttt{B}, \#\texttt{B093})\}$, which is a subset of tuples in `BReach` at peer A corresponding to peer B. Peer C also receives the sub-query and a tuple set $\{(\texttt{C}, \#\texttt{C128})\}$.

In the next step, peer B may, for example, attempt to execute the given sub-queries. If peer B is the originator of the record, the query reaches a *fixpoint* and the resulting relation `Origin` will have a tuple `(B)`, and the result is then returned to peer A. Otherwise, peer B forwards the sub-query with its partial result to the descendant peers on the path to the originator. Peer C also performs similar processing.

Collection of Results As shown above, the given query is processed in a recursive manner in the P2P network. Peer A ultimately receives the results of all the recursive processing via peers B and C. These results are merged and presented to the user as the final result.

The query processing strategy shown above is an extension of the *semi-naive evaluation* strategy in deductive databases [2, 9]. The approach of extending the strategy to distributed networks was originally presented in a *declarative networking* project [8]. Although our approach is a variation of the method presented in [8], our query processing strategy has some differences in the framework which reflect the three-layer organization of record exchange and the structure of logical/physical relations for representing the information used for tracing.

5 Related Work

There are a variety of research topics regarding P2P databases, such as coping with heterogeneities, query processing, and indexing methods [1]. In this paper, we provided a different viewpoint to the research field. The proposed approach is based on the requirement for reliable information exchange in P2P networks. One of the features of our approach is to employ database technologies as the underlying foundation with which to support reliable P2P record exchange.

One related research field is data provenance. The term *data provenance*, or alternatively *lineage tracing*, previously referred to the process of tracing and recording the origins of data and its movement between databases [5, 10]. The target field of data provenance is comparatively wide and covers data warehousing [4], uncertain data management [11], and other scientific fields such as bioinformatics [3]. However, the notion of data provenance has not previously been applied to P2P information exchange to the best of the authors' knowledge.

Some taxonomies have been produced regarding data provenance. [6] presents the notions of *where-provenance* and *why-provenance*. Since our framework treats problems such as where data came from and whether data was reproduced by other peers, it belongs under the heading where-provenance. Another taxonomy distinguishes between the *lazy* approach and the *eager* approach [10]. The former describes models in which queries tracing lineage are executed when necessary and the latter describes the case that metadata and/or annotations [3] representing lineage are maintained. Our approach to traceability is based on histories maintained at peers and thus belongs to the eager approach.

Another related field is *dataspace management* [7]. This is an emerging new research field in the area of databases and focuses on more flexible information integration over the network in an incremental, "pay-as-you-go" fashion. Our approach to P2P record exchange can also be seen as an extension of the traditional approach of data integration: peers in a P2P network can behave autonomously and exchange information when required. In this sense, our information integration framework is quite flexible, but the framework includes traceability functions which yield reliable record exchange.

Our query processing approach is a variation of *declarative networking* as described in [8]. In contrast to their approach, which focuses on efficient query processing in a network (e.g., a sensor network), our target is to represent tracing queries in a compact and clear manner. Since our framework shares the requirement for efficient query processing with their approach, it will be possible to extend our query processing method by considering this former work.

6 Discussions and Conclusions

In this paper, the concept of traceable P2P record exchange was presented and a framework implementing it was shown. The proposed framework consists of three layers, and tracing queries are written in datalog and executed in a recursive manner. We need to consider several issues for the practical implementation of our ideas, for example:

- *Joining/Leaving Facilities*: A dynamic P2P network should provide facilities for joining and leaving for peers, and the peers in the network need to preserve the consistency of the information contained in the network in a cooperative way. The problem is more serious for the leaving facility. One solution would be as follows: 1) the leaving peer A selects a voluntary peer B which can take over its data, 2) peer B copies all the data maintained by peer A, 3) peer B replies to all the subsequent queries on behalf of peer A. We assume that all peers will obey the joining/leaving protocols.
- *Search Facility*: For the search processing, we can apply the existing approaches for efficient P2P search. Some types of the P2P search methods focus on anonymity while exchanging information. In contrast, our method needs search function that explicitly preserves identities of peers which provide the records. That may require slight changes to existing search methods.

In addition, the following issues will be addressed in future work:

– Enhancement of the tracing facilities: It is intended to extend the tracing query language to represent more detailed information.
– Development of efficient query evaluation and optimization techniques: As described above, we will further develop strategies considering the existing methods for deductive databases and P2P databases.
– Prototype system implementation and experiments: We are planning to develop a prototype system for P2P record exchange over relational database management systems (RDBMSs). For this purpose, it is necessary to translate queries in datalog into SQL queries for execution using an RDBMS. It is also necessary to implement facilities for searching, registration, and modification of records.

7 Acknowledgements

This research was partly supported by a Grant-in-Aid for Scientific Research (19300027) from the Japan Society for the Promotion of Science (JSPS). In addition, this work was supported by grants from the Kayamori Foundation of Informational Science Advancement and the Hoso Bunka Foundation.

References

1. K. Aberer and P. Cudre-Mauroux. Semantic overlay networks. In *VLDB*, 2005. (tutorial notes).
2. S. Abiteboul, R. Hull, V. Vianu. *Foundations of Databases*, Addison-Wesley, 1995.
3. D. Bhagwat, L. Chiticariu, W.C. Tan, and G. Vijayvargiya. An annotation management system for relational databases. In *Proc. VLDB*, pp. 900–911, 2004.
4. Y. Cui and J. Widom. Lineage tracing for general data warehouse transformations. In *Proc. VLDB*, pp. 471–480, 2001.
5. P. Buneman, S. Khanna, and W.-C. Tan. Data provenance: Some basic issues. In *Proc. Intl. Conf. of Foundations of Software Technology and Theoretical Computer Science*, Vol. 1974 of LNCS, pp.87–93, 2000.
6. P. Buneman, S. Khanna, and W.-C. Tan. Why and where: A characterization of data provenance. In *Proc. ICDT*, Vol. 1973 of LNCS, pp.316–330, 2001.
7. A. Halevy, M. Franklin, and D. Maier. Principles of dataspace systems. In *Proc. ACM PODS*, pp.1–9, 2006.
8. B.T. Loo, T. Condie, M. Garofalakis, D.E. Gay, J.M. Hellerstein, P. Maniatis, R. Ramakrishman, T. Roscoe and I. Stoica. Declarative networking: Language, execution and optimization. In *Proc. SIGMOD*, pp.97–108, 2006.
9. R. Ramakrishnan and J. Gehrke. *Database Management Systems*. McGraw-Hill, 3rd edition, 2002.
10. W.-C. Tan. Research problems in data provenance. *IEEE Data Eng. Bull.*, 27(4):45–52,2004.
11. J. Widom. Trio: A system for integrated management of data, accuracy, and lineage. In *Proc. CIDR*, pp.262–276, 2005.