“Pay-as-you-go” Processing for Tracing Queries in a P2P Record Exchange System

Fengrong Li[1], Takuya Iida[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,iida}@db.itc.nagoya-u.ac.jp, ishikawa@itc.nagoya-u.ac.jp

Abstract. In recent years, data provenance or lineage tracing has become an acute issue in the database research. Our target is the data provenance issue in peer-to-peer (P2P) networks where duplicates and modifications of data occur independently in autonomous peers. To ensure reliability among the exchanged data in P2P networks, we have proposed a reliable record exchange framework with tracing facilities based on database technologies in [5, 6]. The framework is based on the “pay-as-you-go” approach in which the system maintains the minimum amount of information for tracing with low maintenance cost and a user pays the cost when he or she issues a tracing query to the system. This paper focuses on its two alternative query processing strategies and compare their characteristics according to the performance.

1 Traceable P2P Record Exchange System

Data provenance is a facility that helps database users to interpret database contents and enhances the reliability of data [2, 3]. We focus on the data provenance issue in information exchanges in *peer-to-peer* (*P2P*) networks. Although there exist many P2P systems and related proposals, they do not support the notion of data provenance. Based on this background, we proposed the concept of a *traceable P2P record exchange system* in [5, 6], in which tuple-structured *records* are exchanged in a P2P network. The system employs “pay-as-you-go” approach [4] for tracing. In this paper, we show two alternative query processing strategies and compare their properties.

As an example, assume that information about novels are shared among peers in a P2P network and that each peer maintains a `Novel` record set that has two attributes `title` and `author`. Each peer maintains its own records and keeps its historical data related to itself. To make the tracing process easy, the system provides an abstraction layer which virtually integrates all the distributed relations and a datalog-like query language for writing tracing queries in an intuitive manner. For the details, please refer to [5, 6]. In the following, we present some examples of 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 this record:

```
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 `Data[Novel]` is used to represent the user-level record set `Novel` in the underlying system level (it is called the *logical layer*). A `Novel` record is embedded as the first two attribute values of a `Data[Novel]` tuple. The third attribute of `Data[Novel]` contains the peer name which actually manages the record, and the fourth attribute represents the record id, which is unique among the P2P network. Relation `Exchange[Novel]` represents the record exchange history. For example, a `Exchange[Novel]` tuple `('B', 'A', '#B001', '#A001', '3/2/08')` means that peer A copied a record from peer B, where it had the id value `#B001`, and peer A assigned a new id `#A001` for the record when it was registered at peer A. The last attribute value `3/2/08` represents the timestamp of the exchange.

Query 2: This query detects whether peer C copied the record `(t1, a1)` owned by peer B or not:

```
Reach(P, I1) :- Data[Novel]('t1', 'a1', 'B', I2),
                Exchange[Novel]('B', P, I2, I1, _)
Reach(P, I1) :- Reach(P1, I2), Exchange[Novel](P1, P, I2, I1, _)
Query(I) :- Reach('C', I)
```

Note that Queries 1 and 2 perform backward and forward traversals of provenance information, respectively.

2 Query Processing

Although we have decided to take the "pay-as-you-go" approach and pay the cost when we perform tracing queries, the efficiency of query processing is still a quite important factor. Query 1 can be easily executed using the seminaive strategy [5]; we omit the details here. In the following, we use Query 2 to compare the performance of the *seminaive method* and the *magic set method*.

2.1 Query Evaluation Based on Seminaive Method

The *seminaive method* is based on simple iterative processing, but ensures no redundant evaluations are performed to process a recursive datalog query [1]. A tracing query in our P2P record exchange framework is executed by the cooperation of distributed peers using query forwarding.

Consider Query 2 is issued at peer B. At first, we need to translate it into a query in the *physical layer*. In the physical layer, two *virtual* relations

Data[Novel] and Exchange[Novel] in the logical layer are stored as actual relations in a distributed manner. Each peer only stores its corresponding parts of the logical relations. For example, Data[Novel]'B' relation of the physical layer maintains a subset of Data[Novel] relation in the logical layer which corresponds to peer B. Relation Exchange[Novel] in the logical layer is represented as two physical relations To[Novel] and From[Novel]. For example, To[Novel]@'B' (From[Novel]@'B') represents the information of the records provided (copied) by peer B. In the above query, we used the notation like To[Novel]@P, in which P is a peer variable. The transformation result is as follows.

```
Reach(P, I1) :- Data[Novel]@'B'('t1', 'a1', I2),
                To[Novel]@'B'(I2, P, I1, _)
Reach(P, I1) :- Reach(P1, I2), To[Novel]@P1(I2, P, I1, _)
Query(I) :- Reach('C', I)
```

After the transformation, the query is executed using the extension of the seminaive method. For Query 2, the seminaive method generally visits all the peers which copied the record (t1, a1) offered by peer B. Since record provided by a certain peer often copied by multiple peers, it should visit a number of peers. If we assume that the target record provided by a peer is copied by n peers and m forwarding are performed along every path started from peer B, the process should visit n^m peers in total.

2.2 Query Evaluation Based on Magic Set Method

The *magic set* technique is a well-known strategy for the efficient execution of datalog programs [1]. By modifying a given program, it simulates "selection pushdown" for the top-down evaluation approach within the bottom-up evaluation approach.

First, we transform Query 2 into the following query according to the magic set rewriting rules.

```
Reach(P, I1) :- magic_Reach(P, I1), Data[Novel]@'B'('t1', 'a1', I2),
                From[Novel]@P(I1, 'B', I2, _)
Reach(P, I1) :- magic_Reach(P, I1), Reach(P1, I2),
                From[Novel]@P(I1, P1, I2, _)
magic_Reach(P1, I2) :- magic_Reach(P, I1), From[Novel]@P(I1, P1, I2, _)
magic_Reach('C', I):-
Query(I) :- Reach('C', I)
```

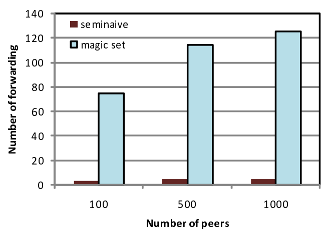
Once a program is modified by the magic set-based rewriting, we can execute the program using the seminaive method. The behavior of the modified program is, however, quite different from the normal seminaive method. In this case, the fourth rule above defines the actual start point; it first triggers the evaluation of the third rule. The additional magic predicate magic_Reach requires the following: for each record in peer C, we should traverse the path from peer C to the origin of the record.

We roughly estimate the cost of the query. Assume that peer C has l records. For each record in peer C, we need to traverse its path to the source. Since we follow the path towards the ancestor, the path does not contain branches. If we assume that the path length is a constant value on average, the total fowarding cost would be $O(l)$. This simple analysis appeals that the magic set-based strategy would be a promising method for Query 2.

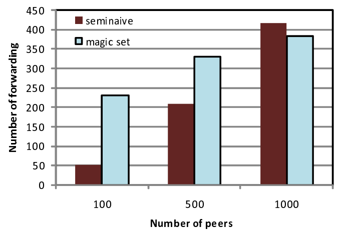
3 Experimental Results

The purpose of the experiments is to observe the behaviors of two query processing strategies using a simple P2P record exchange model. The simulation model is summarized as follows. We first create $N = 100$ peers and $M = 500$ records; each record is randomly assigned to one of the peers. We assume that records are consists of two classes — "hot" records (20%) and normal records (80%). Hot records are more likely to be exchanged; when a peer wants to get a record from other peer, a hot record is selected with the chance of 80%. We perform random record exchanges until each peer exchanges $L = 50$ records on average.

Figure 1 shows the result. In this figure, we added the experimental results for $N = 500$ and $N = 1000$. Their parameters are same except for N.



(a) Average case



(b) Maximum case

Fig. 1. Query forwarding cost for Query 2

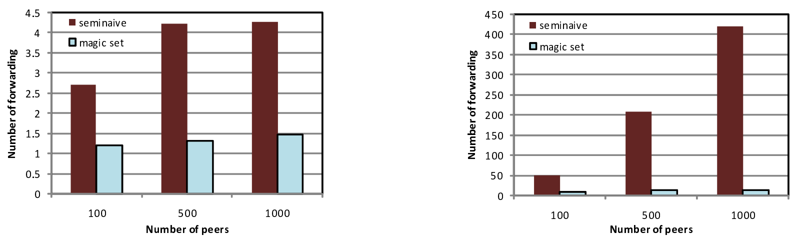
Figure 1(a) and Figure 1(b) shows the same experimental results, the magic set has the high cost. The main reason is that the average number of branches is not high in our simulation model. But the cost of the magic set method becomes better for $N = 1000$. In this case, a hot record is copied by a large number of peers so that the number of branches of forwarding paths become quite large.

Although the magic set has poor performance for the above experiment, it is quite effective for some situations. See the following Query 3, which is a modified version of Query 2.

Query 3: Is the record `(t1, a1)` in peer C a copy of `(t1, a1)` in peer B?

```
Reach(P, I1) :- Data[Novel]('t1', 'a1', 'B', I2),
                Exchange[Novel]('B', P, I2, I1, _)
Reach(P, I1) :- Reach(P1, I2), Exchange[Novel](P1, P, I2, I1, _)
Dup(I) :- Reach('C', I), Data[Novel]('t1', 'a1', 'C', I)
Query(I) :- Dup(I)
```

Figure 2 shows the results. The cost of the seminaive method is same as Query 2. On the other hand, the cost of magic set method becomes quite low, especially in the case of the maximal number of query forwarding. This is because, in contrast to Query 2, the number of forwarding path is only one due to the additional constraint of the third rule used for specifying the start record.



(a) Average case (b) Maximum case

Fig. 2. Query forwarding cost for Query 3

The experimental results indicate that we need to select an appropriate execution strategy depending on the situation.

4 Conclusions

In this paper, we compared two popular query processing methods, the seminaive method and the magic set method for our P2P record exchange framework by experiments: both methods have pros and cons; an appropriate execution strategy depends on the given query, the P2P network organization, the record exchange behaviors, etc. For the long version of this paper, please visit our homepage `http://www.db.itc.nagoya-u.ac.jp/`.

This research was partly supported by the Grant-in-Aid for Scientific Research (19300027, 19024037, 18200005) from MEXT and JSPS.

References

1. S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases.* Addison-Wesley, 1995.
2. P. Buneman, J. Cheney, W.-C. Tan, and S. Vansummeren. Curated databases. In *Proc. ACM PODS*, pp. 1–12, 2008.
3. P. Buneman and W.-C. Tan. Provenance in databases (tutorial). In *Proc. ACM SIGMOD*, pp. 1171–1173, 2007.
4. A. Halevy, M. Franklin, and D. Maier. Principles of dataspace systems. In *Proc. ACM PODS*, pp. 1–9, 2006.
5. F. Li, T. Iida, and Y. Ishikawa. Traceable P2P record exchange: A database-oriented approach. *Frontiers of Computer Science in China*, 2(3):257–267, 2008.
6. F. Li and Y. Ishikawa. Traceable P2P record exchange based on database technologies. In *Proc. APWeb*, Vol. 4976 of *LNCS*, pp. 475–486, 2008.