Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®

Information Processing Letters 89 (2004) 25–29

Information Processing Letters

www.elsevier.com/locate/ipl

Irrelevant updates and self-maintainability in transitive closure database views

Millist W. Vincent *, Jixue Liu

School of Computer and Information Science, The University of South Australia, Mawson Lakes SA5095, Adelaide, Australia

Received 14 April 2003; received in revised form 11 September 2003

Communicated by J. Chomicki

Abstract

Irrelevant updates in a database are updates to source data that do not affect a view defined over the source data. Self-maintainable updates are ones for which the view can be updated, without having to access source data, when the source data from which the view is derived is updated. In this paper we derive necessary and sufficient conditions for an update to be irrelevant or self-maintainable when the source data is a graph and the view is defined to be the transitive closure of the graph.

Keywords: Databases; Irrelevant updates; Self-maintainability; View maintenance; Graphs; Transitive closure

1. Introduction

In this paper we investigate the notions of irrelevant updates and self-maintainability for transitive closure views (graphs which are the transitive closure of a graph) defined over directed graphs. While the framework and results for this paper are presented in a graph theoretic setting, our motivation is from the database area. The notion of an irrelevant update [2,5, 12] is one for which an update to the source database from which a view is defined does not change the view. Detecting such irrelevant updates has application in query optimization [12], data warehousing [13] and data integration [11]. The notion of a view being self-maintainable [2,8,6,7,9,10] is one where the view can be updated, without having to access the source database, when the source database is updated. Self-maintainable views are important in data warehousing applications [6] where the source data is often remote from the view and so being able to update a view without accessing the source data has significant cost and efficiency advantages. Clearly the notions of irrelevant updates and self-maintainable views are related in that if an update is irrelevant to a view then the view is also self-maintainable with respect to the update. However, both concepts are independently useful and so we investigate both in this paper.

In this paper we address the following problems. Given the database as a directed graph and a view defined as the transitive closure of the graph, firstly determine necessary and sufficient conditions under which the insertion or deletion of an edge from the graph is irrelevant to the view. Secondly, determine conditions under which it is possible to update the

* Corresponding author.
E-mail address: millist.vincent@unisa.edu.au (M.W. Vincent).

0020-0190/$ – see front matter
doi:10.1016/j.ipl.2003.09.008

view without accessing the source graph when edges are inserted or deleted in the source graph, i.e., determine conditions under which the view is self-maintainable with respect to the update. We provide necessary and sufficient conditions for all cases.

These results have relevance in relation to some work in [1], which investigated the complexity of answering queries using views. In [1] it was shown that if a view is defined by an arbitrary DATALOG query, then it is undecidable to determine if the view is self-maintainable with respect to insertions or deletions in the underlying database. Our results show that if the view is defined as the transitive closure of a graph, well known to be expressible in DATALOG, then determining if a view is self-maintainable with respect to an update is decidable. Hence it raises the interesting question of detecting where the dividing line is in DATALOG between those views for which determining self-maintainability is decidable and those for which it is not.

2. Related work

The notions of irrelevant updates and self-maintainability were introduced in [2] where they addressed the problem of determining when updates to SPJ (selection-projection-join) views are irrelevant or self-maintainable. The question of whether an update to a view defined in first-order logic is irrelevant was addressed in [5]. In this paper it was shown that if the update itself is allowed to be expressible as a first-order query then the problem of determining if the update is irrelevant to the view is undecidable. In [12], results were derived on irrelevant updates for restricted classes of first-order queries and for updates that are expressible as sets of tuples.

The question of determining whether views expressible as SPJ queries are self-maintainable was also investigated in [6]. However this work differed from the work of [2] in that [6] assumed that additional partial data, such as the view instance or selected base data, is also available. The self-maintainability of SPJ views was also investigated in [10] using a different approach to the one used in [6]. In [10] it was shown that testing for self-maintainability could be made more efficient by splitting the test into a component that is generated at compile-time and a component that is generated at run-time. This approach was later extended in [9] for multiple views. The issue of how to update the transitive closure of a graph in response to updates in the underlying graph was investigated in [4,3], but not from the perspective of irrelevant updates or self-maintainability.

3. Basic definitions and results

A graph $\boldsymbol{G}$ is a pair $(\boldsymbol{V}, \boldsymbol{E})$ where $\boldsymbol{V}$ is a set of vertices and $\boldsymbol{E}$ is a set of edges defined as an ordered relation defined on $\boldsymbol{V} \times \boldsymbol{V}$. Note that in this paper we consider directed graphs.

A graph $(\boldsymbol{V}_1, \boldsymbol{E}_1)$ is a subgraph of $(\boldsymbol{V}, \boldsymbol{E})$ if $\boldsymbol{V}_1 \subseteq \boldsymbol{V}$ and $\boldsymbol{E}_1 \subseteq \boldsymbol{E}$. We denote by $\boldsymbol{G}_1 \subseteq \boldsymbol{G}_2$ the fact that $\boldsymbol{G}_1$ is a subgraph of $\boldsymbol{G}_2$. Two graphs $\boldsymbol{G}_1$ and $\boldsymbol{G}_2$ are identical, denoted by $\boldsymbol{G}_1 = \boldsymbol{G}_2$ if $\boldsymbol{G}_1 \subseteq \boldsymbol{G}_2$ and $\boldsymbol{G}_2 \subseteq \boldsymbol{G}_1$.

A path between vertices a and b is set of edges $e_1, \ldots, e_n$ such that a is the initial vertex of edge e_1, b is the final vertex of the edge e_n and for each pair of edges e_i, e_{i+1}, $1 \leqslant i \leqslant n-1$, the final vertex in e_i is the initial vertex in e_{i+1}.

A path is *simple* if it does not contain repeated edges. A cycle is a simple path which starts and finishes at the same vertex.

A graph is *connected* if between every pair of vertices a and b there exists a path, otherwise it is said to be *disconnected*.

The *transitive closure* of a graph $\boldsymbol{G}$, denoted by $T(\boldsymbol{G})$ is the graph defined over the same set of vertices as $\boldsymbol{G}$ such that an edge (a, b) is in $T(\boldsymbol{G})$ iff there is a path between a and b in $\boldsymbol{G}$.

A graph is a *tree* if it is connected and there are no cycles. A *forest* is a set of trees.

A tree $\boldsymbol{G}^{\mathrm{t}}$ is a *spanning tree* for a connected graph $\boldsymbol{G}$ if it is a subgraph of $\boldsymbol{G}$ which is a tree and contains all the vertices of $\boldsymbol{G}$. Next we present some basic results. We omit the proofs of these results since they are either trivial or can be found in any standard text on graph theory.

Lemma 1. *For any graph $\boldsymbol{G}$, $\boldsymbol{G}$ is a subgraph of $T(\boldsymbol{G})$.*

Lemma 2. *If $\boldsymbol{G}_1$ is a subgraph of $\boldsymbol{G}_2$ then $T(\boldsymbol{G}_1)$ is a subgraph of $T(\boldsymbol{G}_2)$.*

Lemma 3. *If G is a tree then the removal of any edge from G results in a graph that is disconnected.*

Lemma 4. *Let G be a connected graph. Then for any edge $e \in G$ there exists a spanning tree for G which contains e.*

Lemma 5. *Let G be an arbitrary graph. Then for any edge $e \in G$ there exists a subgraph G^f which is a forest and for which $T(G) = T(G^f)$ and for which one of the trees in G^f contains e.*

4. Irrelevant updates

In this section we define and investigate two notions of irrelevant updates. In the first, called global irrelevance, we assume that the only information available is the update itself. In the second, motivated by its application to data warehousing [13,6], we define a more restricted form of irrelevance called local irrelevance. In local irrelevance we assume that the view instance (the transitive closure of some graph) is also available to detect irrelevance.

Before presenting the main definitions, we firstly define legal updates. The insertion of an edge e into a graph G is said to be *legal* if $e \notin G$. The deletion of an edge e from a graph G is said to be *legal* if $e \in G$. Also, we denote by G_μ the graph after a legal update μ (either an insertion or deletion) to a graph G.

Definition 1. Let G be a graph and let $T(G)$ be its transitive closure. An update μ (either the insertion or deletion of an edge e) is *globally transitively irrelevant* if for all G such that μ is legal for G, $T(G) = T(G_\mu)$.

Theorem 1. *Both the insertion and deletion of an arbitrary edge e are not globally transitively irrelevant.*

Proof. Consider first the insertion of an edge e and let G be the empty graph. Clearly the insertion is legal. Then it follows immediately from the definition of transitive closure that $T(G) = \phi$ and $T(G_\mu) = \{e\}$ and so $T(G) \neq T(G_\mu)$ as required. For a deletion the argument is the same except take the initial graph G to be the graph containing the single edge e. □

Definition 2. Let G_1 be a graph and let $T(G_1)$ be its transitive closure. An update μ to G_1 (either the insertion or deletion of an edge e) is *locally transitively irrelevant* to the transitive closure view C, a graph which is the transitive closure of a graph, if for all G such that $C = T(G)$ and μ is legal for G, then $T(G) = T(G_\mu)$.

We now illustrate the definition by an example.

Example 1. Let $C = \{(a, b), (c, d)\}$. Then it can easily be confirmed that the only graph G for which $C = T(G)$ is the graph G where $G = C$. If one considers the deletion μ of the edge (a, b) from G then $T(G_\mu) = \{(c, d)\}$ and so $C \neq T(G_\mu)$ and so the deletion is not locally irrelevant to C.

For insertions, we have the following result.

Theorem 2. *Let C be the transitive closure of a graph G. Then the insertion μ of an edge (a, b) into G is locally transitively irrelevant to C iff $(a, b) \in C$.*

Proof. ($\Rightarrow$) We have to show that $T(G) = T(G_\mu)$ if $(a, b) \in C$. Since $G \subseteq G_\mu$, it follows that $T(G) \subseteq T(G_\mu)$ from Lemma 2. To show that $T(G_\mu) \subseteq T(G)$, let (a_1, b_1) be an arbitrary edge in $T(G_\mu)$. This implies that there exists a path $e_1, \ldots, e_n$ between a_1 and b_1. If any edge e_i in the path is (a, b), then since $(a, b) \in C$ we can replace any occurrence of (a, b) by another path for which all the edges are in G. By doing this we obtain a path between a_1 and b_1 for which all the edges are in G. Hence $(a, b) \in T(G)$ and so $T(G_\mu) \subseteq T(G)$ which completes the If part.

($\Leftarrow$) We shall show the contrapositive that if $(a, b) \notin C$ then the insertion is not locally irrelevant. Since $G_\mu \subseteq T(G_\mu)$ by Lemma 1, it follows that $(a, b) \in T(G_\mu)$ and so $C \neq T(G_\mu)$ and hence the insertion is not locally irrelevant to C. □

For deletions, we have the following negative result.

Theorem 3. *Let C be the transitive closure of a graph G. Then the deletion μ of an edge (a, b) from G is not locally transitively irrelevant to C.*

Proof. From Lemma 5 there exists a spanning forest G^f containing (a, b). However from Lemma 3, the deletion of (a, b) results in one of the trees in G^f

being disconnected and hence $T(\boldsymbol{G}) \neq T(\boldsymbol{G}^{\mathrm{f}}_{\mu})$ (where $\boldsymbol{G}^{\mathrm{f}}_{\mu}$ represents the graph after the deletion of the edge (a,b) from $\boldsymbol{G}^{\mathrm{f}}$) which was required. □

5. Self-maintainable updates

Firstly we recall the notion of global self-maintainability [2].[1]

Definition 3. An update μ (either the insertion or deletion of an edge e) is said to cause *global self-maintainability of transitive closure* if for all $\boldsymbol{G}^1, \boldsymbol{G}^2$ such that $T(\boldsymbol{G}^1) = T(\boldsymbol{G}^2)$ and μ is legal for $\boldsymbol{G}^1$ and $\boldsymbol{G}^2$, then $T(\boldsymbol{G}^1_{\mu}) = T(\boldsymbol{G}^2_{\mu})$.

For insertions, we have the following result.

Theorem 4. *If the update μ is the legal insertion of an edge e into a graph, then μ causes global self-maintainability of transitive closure.*

Proof. We firstly show that for any graph $\boldsymbol{G}$, $T(\boldsymbol{G}_{\mu}) = T((T(\boldsymbol{G}))_{\mu})$. To show this firstly note that $T(\boldsymbol{G}_{\mu}) \subseteq T((T(\boldsymbol{G}))_{\mu})$ because, by Lemma 1, $\boldsymbol{G} \subseteq T(\boldsymbol{G})$ and hence $\boldsymbol{G}_{\mu} \subseteq (T(\boldsymbol{G}))_{\mu}$ and thus $T(\boldsymbol{G}_{\mu}) \subseteq T((T(\boldsymbol{G}))_{\mu})$ by applying Lemma 2. To show that $T((T(\boldsymbol{G}))_{\mu}) \subseteq T(\boldsymbol{G}_{\mu})$, let (a,b) be an arbitrary edge in $T((T(\boldsymbol{G}))_{\mu})$. This implies that there exists a path $e_1, \ldots, e_n$ between a and b, where each edge is in $T(\boldsymbol{G})_{\mu}$. Then, for each edge e_i in the path such that $e_i \in T(\boldsymbol{G})$, we can replace e_i by another path $e_{i_1}, \ldots, e_{i_n}$. By doing this for each $e_i \in T(\boldsymbol{G})$, we obtain a path between a and b such that each edge in the path is in $\boldsymbol{G}$ or is e. Thus $(a,b) \in T(\boldsymbol{G}_{\mu})$ and so $T((T(\boldsymbol{G}))_{\mu}) \subseteq T(\boldsymbol{G}_{\mu})$ which completes the proof that $T((T(\boldsymbol{G}))_{\mu}) = T(\boldsymbol{G}_{\mu})$.

Consider then any two graphs $\boldsymbol{G}^1, \boldsymbol{G}^2$ such that $T(\boldsymbol{G}^1) = T(\boldsymbol{G}^2)$. Then

$$
\begin{aligned}
T\big(\boldsymbol{G}^1_{\mu}\big) &= T\big(\big(T(\boldsymbol{G}^1)\big)_{\mu}\big) \\
&\qquad \text{from the result just established} \\
&= T\big(\big(T(\boldsymbol{G}^2)\big)_{\mu}\big) \qquad \text{from assumption} \\
&= T\big(\boldsymbol{G}^2_{\mu}\big) \qquad \text{from the result just established}
\end{aligned}
$$

This completes the proof. □

[1] Self-maintainability was referred to as autonomously computable in [2].

For the deletion of an edge, we have the following negative result.

Theorem 5. *The deletion of an edge e from a graph does not cause global self-maintainability of transitive closure.*

Proof. Let $\boldsymbol{G}^1$ be the graph $\{(a,b),(b,c)\}$ and let $\boldsymbol{G}^2$ be the graph $\{(a,b),(b,c),(a,c)\}$. It is easily verified that $T(\boldsymbol{G}^1) = T(\boldsymbol{G}^2)$. Then, after the deletion μ of the edge (a,b), $T(\boldsymbol{G}^1_{\mu}) = \{(b,c)\}$ and $T(\boldsymbol{G}^2_{\mu}) = \{(a,b),(b,c),(a,c)\}$ and so $T(\boldsymbol{G}^1_{\mu}) \neq T(\boldsymbol{G}^2_{\mu})$ and the deletion is not globally self-maintainable. □

Following the approach in [6,10], we now consider self-maintainability in the context where we can use the view instance itself to detect self-maintainability. In our context, this means that the transitive closure of the graph is available.

Definition 4. Let $\boldsymbol{G}$ be a graph and let $T(\boldsymbol{G})$ be its transitive closure. Let μ be either the insertion or deletion of an edge e into $\boldsymbol{G}$. Then the transitive closure view $\boldsymbol{C}$, where $\boldsymbol{C}$ is the transitive closure of some graph, is *locally self-maintainable* with respect to μ if for all $\boldsymbol{G}^1$, $\boldsymbol{G}^2$ such that $T(\boldsymbol{G}^1) = T(\boldsymbol{G}^2) = \boldsymbol{C}$ and μ is legal for $\boldsymbol{G}^1$ and $\boldsymbol{G}^2$, then $T(\boldsymbol{G}^1_{\mu}) = T(\boldsymbol{G}^2_{\mu})$.

We now illustrate the definition by two examples.

Example 2. Let $\boldsymbol{C} = \{(a,b),(b,c),(a,c)\}$. Then it can easily be confirmed that if we let $\boldsymbol{G}^1 = \{(a,b),(b,c),(a,c)\}$ and $\boldsymbol{G}^2 = \{(a,b),(b,c)\}$ then $T(\boldsymbol{G}^1) = T(\boldsymbol{G}^2) = \boldsymbol{C}$. If one considers the deletion μ of (a,b) from $\boldsymbol{G}^1$ and $\boldsymbol{G}^2$ then $T(\boldsymbol{G}^1_{\mu}) = \{(b,c),(a,c)\}$ and $T(\boldsymbol{G}^2_{\mu}) = \{(b,c)\}$ and so $T(\boldsymbol{G}^2_{\mu}) \neq T(\boldsymbol{G}^1_{\mu})$ and so the deletion is not locally self-maintainable.

Example 3. Let $\boldsymbol{C} = \{(a,b),(c,d)\}$. In this case the only database $\boldsymbol{G}$ for which $\boldsymbol{C} = T(\boldsymbol{G})$ is the database $\boldsymbol{G}$ where $\boldsymbol{G} = \boldsymbol{C}$. Hence the deletion of (a,b) is locally self-maintainable and $\boldsymbol{C}$ can be brought up to date by deleting the edge (a,b) from $\boldsymbol{C}$.

Theorem 6. *Let $\boldsymbol{C}$ be the transitive closure of a graph $\boldsymbol{G}$. Then $\boldsymbol{C}$ is locally self-maintainable with respect to the insertion of an edge e into $\boldsymbol{G}$.*

Proof. Follows immediately from Theorem 4 and the fact that global self-maintainability implies local self-maintainability. □

For deletions, we have the following.

Theorem 7. *Let $\boldsymbol{C}$ be the transitive closure of a graph $\boldsymbol{G}$. Then $\boldsymbol{C}$ is locally self-maintainable with respect to the deletion μ of an edge (a, b) from $\boldsymbol{G}$ iff neither a nor b is connected to any other vertex in $\boldsymbol{C}$.*

Proof. ($\Rightarrow$) Let $\boldsymbol{G}$ be any graph such that $\boldsymbol{C} = T(\boldsymbol{G})$. By definition of transitive closure, if neither a nor b is connected to any other vertex in $\boldsymbol{C}$ then the same property will hold in $\boldsymbol{G}$. We claim that to update $\boldsymbol{C}$ we simply remove (a, b) from $\boldsymbol{C}$. To show this, firstly (a, b) must be removed from $\boldsymbol{C}$ since otherwise it would imply that there exists another path from a to b in $\boldsymbol{G}$ which contradicts the fact that neither a nor b is connected to any other vertex in $\boldsymbol{G}$. Secondly, we claim that if (c, d) is any other edge in $\boldsymbol{C}$ then (c, d) will still be in $\boldsymbol{C}$ after the deletion. To see this, if $(c, d) \in \boldsymbol{G}$ then automatically (c, d) will still be in $\boldsymbol{C}$ after the deletion. Suppose instead that $(c, d) \notin \boldsymbol{G}$. Then this implies that there is a path from c to d in $\boldsymbol{G}$. However, because neither a nor b are connected to any other vertex in $\boldsymbol{G}$, the path cannot contain the edge (a, b) and so the path will still be in $\boldsymbol{G}$ after the deletion and so (c, d) will remain in $\boldsymbol{C}$. Hence to update $\boldsymbol{C}$, we simply remove the edge (a, b) from $\boldsymbol{C}$ and since this is clearly independent of $\boldsymbol{G}$ the result is proven.

($\Leftarrow$) Suppose to the contrary that there is another edge (b, c) in $\boldsymbol{C}$ (the same argument applies if a is the common vertex). Then it follows from the proof of Lemma 4 that (a, b) and then (b, c) can be chosen to be the first two edges to be added to the spanning forest and hence will remain in the spanning forest. Let's denote this spanning forest by $\boldsymbol{G}^1$. Consider then the graph $\boldsymbol{G}^2$ obtained by adding the edge (a, c) to $\boldsymbol{G}^1$. Since the edge (a, c) already exists in $T(\boldsymbol{G}^1)$, it follows using similar arguments to those used in Theorems 2 and 4 that $T(\boldsymbol{G}^1) = T(\boldsymbol{G}^2)$. Consider then the effect of removing the edge (a, b) from $\boldsymbol{G}^1$ and $\boldsymbol{G}^2$. In $T(\boldsymbol{G}^1)$, the removal of (a, b) results in (a, c) being removed from $T(\boldsymbol{G}^1)$ since otherwise it would contradict the fact that (a, b) belongs to a spanning tree. In contrast, if we remove (a, b) from $\boldsymbol{G}^2$ then a and c remain connected and so (a, c) remains in the transitive closure. Hence $T(\boldsymbol{G}^1_\mu) \neq T(\boldsymbol{G}^2_\mu)$ which completes the Only If part. □

References

[1] S. Abiteboul, O.M. Duschka, Complexity of answering queries using materialized views, in: ACM PODS Conference, 1998, pp. 254–263.

[2] J. Blakeley, J. Coburn, P.N. Larson, Updating derived relations: Detecting irrelevant and autonomously computable updates, ACM Trans. Database Systems 14 (3) (1989) 269–400.

[3] G. Dong, C. Pang, Maintaining transitive closure in first order after node-set and edge-set deletions, Inform. Process. Lett. 62 (4) (1997) 193–199.

[4] G. Dong, J. Su, Incremental and decremental evaluation of transitive closure by first-order queries, Inform. and Comput. 120 (1) (1995) 101–106.

[5] C. Elkan, Independence of logic database queries and updates, in: Proc. of 9th ACM Symp. on Principles of Database Systems (PODS 1990), Nashville, TN, 1990, pp. 154–160.

[6] A. Gupta, J. Blakeley, Using partial information to update a materialized view, Inform. Systems 20 (8) (1995) 641–662.

[7] A. Gupta, I. Mumick, Maintenance of materialized views: Problems, techniques, and applications, Data Engrg. Bull. 18 (2) (1995) 1–16.

[8] A. Gupta, H.V. Jagadish, I.S. Mumick, Data integration using self-maintainable views, in: Proc. of 5th Internat. Conf. on Extending Database Technology (EDBT 1996), Avignon, France, 1996, pp. 140–144.

[9] N. Huyn, Multiple-view self-maintenance in data warehousing environments, in: Proc. of 23rd Internat. Conf. on Very Large Data Bases (VLDB 1997), Athens, Greece, 1997, pp. 26–35.

[10] N. Huynh, Efficient view self-maintenance, in: ACM Workshop on Materialized Views: Techniques and Applications, 1996.

[11] A. Levy, Obtaining complete answers from incomplete databases, in: Proc. of 22nd Internat. Conf. on Very Large Data Bases (VLDB 1996), Bombay, India, 1996, pp. 402–412.

[12] A.Y. Levy, Y. Sagiv, Queries independent of update, in: Proc. of 19th Internat. Conf. on Very Large Data Bases (VLDB 1993), Dublin, Ireland, 1993, pp. 171–181.

[13] J. Widom, Research problems in data warehousing, in: Proc. of Conference on Information and Knowledge Management (CIKM), Baltimore, MD, 1995, pp. 25–30.