COMPUTING TRANSITIVE CLOSURES OF MULTILEVEL RELATIONS

Bhavani Thuraisingham
The MITRE Corporation, Bedford, MA 01730

ABSTRACT

Recently many attempts have been made to implement recursive queries. Such queries are essential for the new generation intelligent database system applications. Much of the effort in recursive queries is focussed on transitive closure queries which are of practical significance. None of the work described investigates recursive query processing in secure database management systems. In this paper we propose centralized and distributed algorithms for implementing transitive closure queries for multilevel secure relational database management systems.

1. INTRODUCTION

During the past decade much effort was directed towards integrating database and knowledge base systems [BROD86] from which intelligent database systems emerged not only as prototypes [RAMN88] but also as commercial products [COHE89]. The emergence of such systems was a milestone because it enabled database systems to perform deductive reasoning instead of being merely repositories of data. The deductive reasoning process in many instances can be generated by specifying the request as a single recursive query. Otherwise the deductive process has to be generated by a sometimes cumbersome set of nonrecursive queries or even by a lengthy application program. In order to process recursive queries many efficient algorithms including parallel ones have been developed. [See for example BANC86, AGRA87, VALD88]. Although numerous articles have been written on the concept and implementation of recursive queries, no attempt has yet been made to incorporate them in secure data/knowledge base systems work. This paper makes an initial effort on proposing algorithms to implement a special type of recursive query, called a transitive closure query, for secure database systems.

For many of the applications in the military environment which use multilevel database systems, there is a need for recursive query processing. For example, it may be necessary to list all the military bases that are reachable from a particular base, or to list all the subparts of a missile. In a multilevel environment, additional controls are necessary so that the responses released can be legitimately read by the querying user. For example, it may be the case that certain bases or routes to certain bases are classified at the Secret level. In such a situation, an Unclassified user should not be able to read the Secret bases or the Secret routes to the bases. This paper proposes recursive query processing algorithms which ensure that a user only gets the responses that he is authorized to read.

It was proved in [AHO79] that a query to list all the points that are reachable from a particular point cannot be expressed using standard relational algebraic operators proposed by Codd [CODD70]. Such a query is an example of a recursive query. It was also shown that the expression of such a query requires an additional operator, called a least fixed point operator. Queries that are obtained by using the least fixed point operator are fixed point queries. Although Codd states that the relational algebra is computationally complete, Aho and Ullman have shown that it is necessary to augment relational algebra with the least fixed point operator to obtain computational completeness. The fixed point queries are a special type of recursive queries. It was shown in [CHAND82] that all practically computable meaningful queries fall into the category of fixed point queries.

A special type of fixed point query that has received considerable attention over the past decade is the transitive closure query which requests to compute the transitive closure of a relation.

Examples of requests which fall in the category of transitive closure queries are:

- Find all pairs of cities (x,y) where y is reachable from x.
- Find all pairs of persons (x,y) where x is an ancestor of y.
- Find all pairs of employees (x,y) where y is subordinate to x.
- Find all pairs of parts (x,y) where y is a subpart of x.

Note that all of these queries are recursive. This is because the relation to be computed is defined in terms of itself. For example, in the case of ancestor relationship one has the rules:

- A parent is an ancestor,
- An ancestor of an ancestor is also an ancestor.

That is, the ancestor relationship is defined in terms of itself.

In this paper we propose algorithms for computing the transitive closure of centralized and distributed, multilevel relations. These algorithms are an adaptation of the transitive closure algorithms developed in [VALD88] for non-multilevel relations. We also give a formalism for expressing the transitive closure of multilevel relations using the fixed point operator. It is important to start work in this area for many reasons. They include the following:

- The applications which currently use multilevel systems such as military and C^3I are also involved in complex computations which may be generated by recursive queries,
- Such work may result in closer collaboration between the database and security researchers, which is essential if useful database security products are to be developed,
- Such work will ultimately produce secure intelligent database management systems.

The organization of this paper is as follows:

In section 2 we define multilevel relations. In section 3 we state the concepts on fixed point queries and transitive closures of relations. In section 4 we discuss the notion of transitive closures of multilevel relations. In section 5 we describe a technique for enforcing a necessary integrity rule on security constraints. In section 6 we give algorithms for computing the transitive closures of multilevel relations. The paper is concluded in section 7.

2. MULTILEVEL RELATIONS

Various definitions of a multilevel relation have been proposed (see for example [GRAU82, DENN87, STAC89]). For the discussion in this paper, we assume that in a multilevel relation it is possible for the attributes, elements and tuples to be assigned different security levels. The security level assigned to each component of a relation must dominate the security level assigned to the relation. If an attribute (or a tuple or an element) is assigned the Unclassified security level, then the relation which contains it cannot be assigned the Secret level. The security level assigned to an element must dominate the security level of the attribute to which the element belongs to. For example, if the security level of an attribute is Secret, then all of the elements of that attribute must be at least Secret. Tuple classifications assign security levels to the relationship between the elements of a tuple. Therefore, it is possible for the individual elements of a tuple to have a lower security level than that of the tuple. We assume that security levels are assigned to the various components of a relation using security constraints.

Multilevel Relation EMP

	Name	Salary	SS#	Department
	N1-U	S1-S	T1-U	D1-U
	N2-S	S2-S	T2-U	D2-U
	N3-U	S3-S	T3-U	D3-S
TS Tuple	N4-U	S4-S	T4-U	D4-U

Figure 1. Multilevel Relation

An example of a multilevel relation is illustrated in figure 1. In this figure, relation EMP with attributes SS#, Name, Salary and Department is classified at the Unclassified level. The salary attribute is Secret and therefore all the salary values are Secret. The elements N2 and D3 are Secret. The tuple which describes the employee N4 is Top Secret. Therefore, this tuple cannot be seen by a user at a lower level. Note that it is possible for lower level users to see parts of this tuple and infer the Top Secret information. Therefore additional controls are necessary to prevent such security violations.

Relational algebraic operators such as select, project, join, cartesian product, union and difference can be applied to multilevel relations. A possible approach to applying these operators to multilevel relations is given below.

Consider the multilevel relation R. Then (R,L), the restriction of R at L, is the relation R at security level L and is obtained as follows:

(i) Obtain R* from R by eliminating all tuples whose security levels dominate L.
(ii) For each element a in R*, if the security level of a is dominated by L, then a is also in (R,L). If not, a is replaced by a variable in (R,L) (note that the same variable is used to replace all occurrences of a).

The commonly used operators Select, Project and Join can be defined on multilevel relations as follows. We assume that the application of these operators are requested by a user at security level L.

$\text{Select}_{\text{cond and } L}(R) = (S, L)$ where S is $\text{Select}_{\text{cond}}(R)$
$\text{Project}_{\text{attr. and } L}(R) = (S,L)$ where S is $\text{Project}_{\text{attr}}(R)$
$\text{Join}_{\text{attr and } L}(R1, R2) = \text{Select}_{\text{attr}}(S, L)$ where S is Product(R1,R2)

A select operation on R under condition 'cond' at security level L is the relation S at security level L (i.e. , the relation (S,L)) where S is the result of applying the select operation under condition 'cond' on R.

A project operation on attributes 'attr' of R at security level L is the relation (S,L) where S is the result of applying the project operation on attributes 'attr' of S.

A join operation on R1 and R2 on attributes 'attr' at security level L is obtained by first computing the product S of R1 and R2, then forming (S,L) and finally doing a selection on (S,L) such that the values of the join attributes 'attr' are the same in (S,L).

A multilevel database is a database which consists of multilevel relations. A trusted database management system (TDBMS), also called a multilevel secure database management system (MLS/DBMS), manages a multilevel database.

3. FIXED POINT QUERIES AND TRANSITIVE CLOSURES

In this section we state the definitions of least fixed point operator and transitive closures. Much of the information presented in this section has been obtained from [AHO79].

Consider the equation S = f(S) where f(S) is the relational algebraic expression which results from applying any function f to the relation S such that the degree of S and f(S) are the same. A least fixed point of the equation above, denoted by LFP(S = f(S)) is a relation S* which satisfies the following two conditions:

(i) S* = f(S*)

(ii) If Q is any relation such that $Q = f(Q)$, then S^* is a subset of Q (where the relation X is a subset of the relation Y if every tuple which belongs to X also belongs to Y).

Aho and Ullman have shown that there is a function f^* which is monotone (that is, if the relation X is a subset of the relation Y, then $f^*(X)$ is a subset of $f^*(Y)$) and therefore by the results obtained by Tarski [TARS55] on fixed point theory of lattices, there is a unique least fixed point for the equation $S = f^*(S)$. In the definition of the transitive closure of a relation to be given below, it will be seen that $S = f^*(S)$ is the equation $S = S \cdot R \cup R$ and that for every R, the least fixed point of this equation is the transitive closure of R.

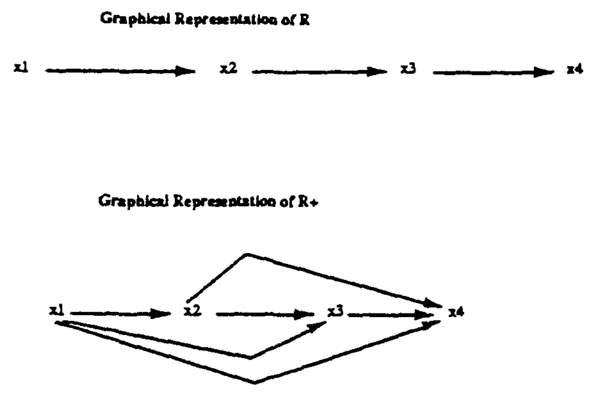

Figure 2. Transitive Closure of a Relation

In order to define the transitive closure of a relation, first we need to define the composition operator.

Given the binary relations R and S where their attributes take values in a single domain, the composition operation is defined as follows:

$$R \cdot S = \{(x,y) \mid \exists z\ (x,z) \in R \text{ and } (z,y) \in S\}.$$

Note that the composition operation can be implemented by a join with a projection.

The transitive closure of a binary relation R, denoted by R+, is defined as follows:

$$R+ = \bigcup_{i > 0} R^i \text{ where}$$

$$R^1 = R \text{ and } R^i = R^{i-1} \cdot R.$$

The transitive closure of a relation R can be expressed using the least fixed operator LFP as follows:

$$R+ = LFP(S = S \cdot R \cup R)$$

where U is the union operator and • is the composition operator.

That is, the least fixed point of the equation $S = S \cdot R \cup R$ is the transitive closure of the relation R.

We illustrate the transitive closure of a relation with an example:

Consider the relation R with tuples (x1,x2), (x2,x3) and (x3,x4). The transitive closure of R, denoted by R+, consists of the following tuples:

(x1,x2), (x2,x3), (x3,x4), (x1,x3), (x2,x4), (x1,x4).

The graphical representations of R and R+ are shown in figure 2.

4. TRANSITIVE CLOSURES OF MULTILEVEL RELATIONS

In this section we describe the policy for computing transitive closures of multilevel relations. Consider a relation R which has (x,y) and (y,z) as some of its elements. Then the pair (x,z) belongs to R+. However, if R is a multilevel relation, whether the pair (x,z) belongs to R+ depends on the security levels assigned to (x,y), (y,z), x,y and z.

Suppose there are only two security levels: Secret and Unclassified. Also assume that an Unclassified user has requested the computation of the transitive closure of R. We assume the following in this computation:

(i) If (a,b) belongs to R and a,b, (a,b) are all Unclassified, then (a,b) is included in the computation of the transitive closure of R at level Unclassified.

(ii) If (a,b) belongs to R and (a,b) is Secret (note that a, b could be either Unclassified or Secret), then (a,b) is not included in the computation of the transitive closure of R at level Unclassified (i.e., as far as the Unclassified user is concerned (a,b) does not exist).

(iii) If (a,b) belongs to R and one or both of a,b are Secret, then the Secret item is replaced by a variable (the same variable is used to replace all occurrences of this item) and the resulting pair is included in the computation of the transitive closure of R at the level Unclassified.

The security policy for computing the transitive closure of a relation R is as follows: Suppose a user at a security level L requests to compute the transitive closure of a relation R, denoted TC[L,R].

We define TC[L,R] = ((R,L)+, L).

That is, the following steps are involved in the computation of the transitive closure of a multilevel relation R at a security level L.

(i) Compute (R,L)
(ii) Compute (R,L)+
(iii) Compute ((R,L)+, L).

Using the least fixed point operator (described in section 3), the transitive closure of a multilevel relation R at a security level L is expressed by:

$$TC[L,R] = (LFP(T = f(T) \cup (R,L)), L)$$
where $T = S$ and $f(T) = (S \cdot (R,L))$.

Note that step (iii) in the computation (of the transitive closure of a multilevel R) will not be necessary if the security constraints on the tuples are correctly enforced. In order to correctly enforce the security constraints, an integrity rule must be satisfied. In section 5 we define this integrity rule and state an algorithm which enforces this rule.

We illustrate the transitive closure of a multilevel relation R with an example.

Consider the relation R shown in figure 2. Let the following constraints be enforced; (x1,x4) is Secret, (x2,x4) is Secret, and (x3,x4) is Secret . In order to compute the transitive closure of R at level L, we first compute (R,L) and then compute (R,L)+. The computation is shown in the following steps:

(i) R = {(x1,x2), (x2,x3), (x3,x4)}
(ii) (R,L) = {(x1,x2), (x2,x3)}
(iii) (R,L)+ = {(x1,x2), (x2,x3), (x1,x3)}
(iv) ((R,L)+,L) = {(x1,x2), (x2,x3), (x1,x3)}

5. ENFORCING AN INTEGRITY RULE ON SECURITY CONSTRAINTS

Suppose a relation R consists of the pairs (x1,x2) and (x2,x3). Assume that (x1,x2) and (x2,x3) are both Unclassified while the pair (x1,x3) which belongs to R+ is assigned a Secret level. Then,

TC[Unclassified, R] = {(x1,x2), (x2,x3)}.

This means an Unclassified user can obtain the pairs (x1,x2), (x2,x3) and infer the Secret pair (x1,x3). In order to prevent such security violations (due to inference [THUR87]), it should be ensured that at least one of (x1,x2), (x2,x3) is Secret. In general the following integrity rule should be enforced on the security constraints:

Integrity Rule: If the pair (x1,xn) (belonging to R+) is classified at the Secret level, then for every path P between x1 and xn, there exist two elements xi, xj which are adjacent in P such that the pair (xi,xj) is classified at the Secret level.

(Note that x1, x2, x3, ...xn-1, xn is a path from x1 to xn if the pairs (x1,x2), (x2,x3),.......... (xn-1,xn) are all members of the relation R.)

In this section we will give an algorithm which enforces the integrity rule on the security constraints. That is, from an initial set of constraints, the algorithm will enumerate a larger set of constraints which have to be enforced in order to satisfy the integrity rule. This algorithm can be used by the system security officer during system initialization.

Note that if the integrity rule is enforced, then it is not necessary to process the third step in the computation of the transitive closure of a multilevel relation R. That is, the computation of ((R,L)+,L) can be skipped. Therefore, TC[L,R] = (R,L)+.

Algorithm IRE (Integrity Rule Enforcement)

```
Begin

    For each multilevel relation R do the following.

        Let C be the list of initial constraints for R.
        A constraint will be specified as a pair. That
        is, if (x1,x2) is in C then the pair (x1,x2) is
        Secret. (Note: we assumed that there are
        only two security levels.)

        Use temporary lists T and L. T is initialized
        to C. L is initialized to the empty set. The
        members of L will be pairs of 3-
        tuples. Each 3 tuple will be of the
        form (element1, element2, Path)

        While T is not empty do the following
        Begin

        Remove a constraint (x1,xn) from T

        Mark (x1,xn) as S

        Find all paths from x1 to xn

        For each path P do the following:

        Check whether there are adjacent elements
        xi, xj in P such that some constraint in C
        classifies the pair (xi,xj) at the Secret level.
        If so, do nothing.

        If not, then mark the pair (xn-1, xn) as S
        (where xn-1 is the element which
        precedes xn in the path P).

        For each element E = (yi, yj, Q) in the list L
        do the following:

        Check to see whether the pair (yi, yj) still
        needs to have a marker S. (Note
        that by marking (xn-1, xn) by S,
        the pair (yi, yj) may not need the
        marker S.)

        If (yi,yj) no longer needs a marker, then
        delete the marker S assigned to it.

        End (for each element)

        Place (xn-1, xn, P) in the list L

        End (for each path P)

        End (while T is not empty)

        Any pair which has the marker S beside it is
        a constraint; Update the list C of constraints
        to include the new constraints

    End (for each multilevel relation R)

End (of algorithm)
```

An Example:

We illustrate this algorithm with an example. Let R be the relation with members (x0,x1), (x1,x2), (x2,x3), (x3,x4). Its transitive closure R+ at the Secret level will have all the members of R plus the following additional members:

(x0,x2),(x2,x4),(x1,x4),(x0,x3)(,(x0,x4),(x1,x3).
Let the initial constraints be (x1,x4) and (x0,x2).

During the first iteration the constraint (x1,x4) will be processed.

At the end of this iteration the pairs (x1,x4) and (x3,x4) will be marked S. Also the triple (x3,x4,P) is placed in the list L where P is the path (x1,x2,x3,x4).

During the second iteration, the constraint (x0,x2) will be processed.

First the pair (x1,x2) will be assigned the marker S. Then the list L is examined. As a result of marking
(x1,x2), it is no longer necessary to mark (x3,x4) by S. Therefore, the marker S is deleted from (x3,x4).

The constraints are (x1,x4), (x0,x2), (x1,x2).

6. TRANSITIVE CLOSURE ALGORITHMS

We propose two algorithms to compute the transitive closure of multilevel relations. The first algorithm is for centralized relations and the second is for distributed relations. The algorithms that we state here are an adaptation of the transitive closure algorithms described in [VALD88] for non-multilevel relations. Many of the other algorithms proposed in the literature can also be adapted for multilevel relations. Describing all of them is beyond the scope of this paper. However, the essential points of the argument can be exhibited with the algorithms that we develop here.

Section 6.1 describes the algorithm for centralized relations and section 6.2 describes the algorithm for distributed relations. In these algorithms, we assume that the integrity rule defined in section 5 is enforced on the security constraints.

6.1 Algorithms for Centralized Multilevel Relations

An architecture for computing the transitive closures of multilevel relations is shown in figure 3. In this architecture, a TDBMS is augmented with a Transitive Closure Processor. The TDBMS is responsible for building a view of the multilevel relation at the security level of the user who requests the transitive closure computation. The security constraints enforced on the tuples are used by the TDBMS to build the views of the relations. It is assumed that the IRE algorithm will generate the necessary constraints from the initial set of constraints specified, say, by the systems security officer during system initialization. The view computed by the TDBMS is given to the Transitive Closure Processor which computes the transitive closure. The Transitive Closure Processor does not have to be trusted (i.e., it does not perform security critical functions) as all of the constraints are handled during the computation of the view.

The view R of a relation at a security level L is (R,L). It is constructed as follows:

(i) If (a,b) is classified at L' where L' dominates L, then (a,b) is not included in the view.

(ii) If one (or both) of a,b is classified at L' where L' dominates L, then this item (or both items) is replaced by a variable (note that the same variable is used for all occurrences of this item). The resulting tuple is placed in the view.

(iii) All the remaining tuples of R are placed in the view (i.e., a, b, and (a,b) are dominated by L).

The Transitive Closure Processor will then compute the transitive closure of the view (R,L). It can use any algorithm in the literature for computing transitive closure of relations. If requested by the user, it could delete those tuples belonging to the closure which contain variables; although displaying the variables will not cause security violation as the user cannot see the value of the variable.

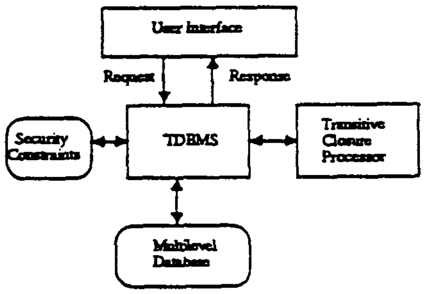

Figure 3. Architecture for Transitive Closure Query Processing

MITC Algorithm (Multilevel Iterative Transitive Closure)

The input is the multilevel relation R and the security level L. The output is T = TC[L,R].

```
Begin

1. P:= (R, L)   /* the TDBMS will compute this*/
2. T := P
3. D := P

Repeat
4. D := D • P
5. T := T U D

until (D = empty) or (D intersection T is empty)

end (algorithm MITC)
```

Example:

Consider the relation R illustrated in figure 2. The initial constraint is (x1,x4). The algorithm to enforce the integrity rule given in the previous section will generate the additional constraint (x3,x4). A trace of MITC given input (R, Unclassified) is given below:

Iteration 1:
1. P := {(x1,x2),(x2,x3)}
2. T := {(x1,x2), x2,x3)}
3. D := {(x1,x2),(x2,x3)}
4. D := {(x1,x3)}
5. T := T U {(x1,x3)} = {(x1,x2),(x2,x3),(x1,x3)}

Iteration 2:
4. D := empty
5. T := {(x1,x2),(x2,x3),(x1,x3)}

Since D is empty, the algorithm terminates.

6.2 Algorithms for Distributed Multilevel Relations

The architecture for computing the transitive closure of distributed relations is shown in figure 4. In this architecture, each node has a Transitive Closure Processor, a TDBMS and a Multilevel Database. The multilevel relations are distributed across the nodes. It is assumed that the network which interconnects the various nodes is multilevel secure.

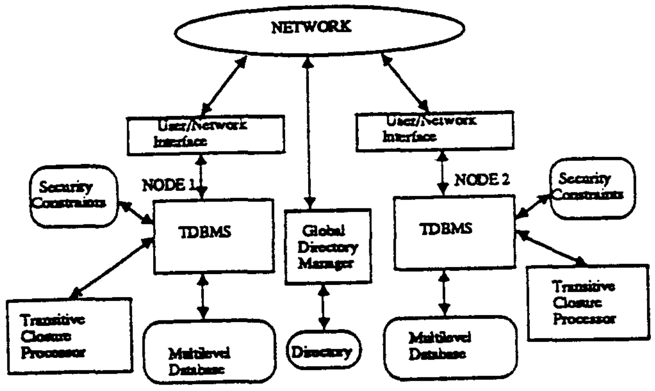

Figure 4. Transitive Closure Processing of a Distributed Relation

In the algorithm to compute the transitive closure of a multilevel distributed relation, each TDBMS computes the view of the portion of the multilevel relation that is stored at its site, at the user's security level (note that if R is the fragment of a relation at a site, then (R,L) is the view of that fragment at security level L). Then all the views computed at the various nodes are sent to a designated site which is responsible for computing the transitive closure. The Transitive Closure Processor at the designated site computes the transitive closure.

We assume that the Transitive Closure Processor at each node is untrusted. Therefore, the TDBMS at each node should have access to the entire list of security constraints in order to compute the views of the multilevel relations. Otherwise, the designated Transitive Closure Processor may also have to examine some constraints in order to eliminate some sensitive tuples and therefore it has to be trusted. If a list of all the security constraints has to be accessed by each TDBMS, then the system security officer should execute the Integrity rule enforcement algorithm on the initial set of constraints during system initialization.

Before sending the views of the multilevel relations to the designated site, the variable reference conflicts have to be resolved. That is, site 1 could use the variable x for a classified element 'a' while site 2 could use the variable y for the same element. One way to resolve the conflicts is for each TDBMS to send the variable name and the actual value of the variable to a trusted process which we call the Global Directory Manager. This Global Directory Manager will determine the variable to use by analyzing the information that it received from each TDBMS. The TDBMS then replaces the variable name that it used by the variable name it received from the Global Directory Manager. The view of the relation is then sent to the designated site.

MTCUP Algorithm (Multilevel Transitive Closure with Unique Processor)

The inputs to this algorithm are the relation R and the security level L. The output is T = TC[L,R].
Assume that there are n nodes. At each node i (1 <= i <= n) the fragment Ri of R is stored. The transitive closure is computed at one node only. Let this node be N.

The algorithm consists of two phases.

```
Phase 1:  at each node i (1 <= i <= n) do

          Compute (Ri, L)
          Resolve variable name conflicts
          Let the result be Si
          Send Si to node N, the designated site.

Phase 2:  at node N do

          Form the union of all the fragments Si
          Let the result be S
          Compute S+
          S+ is the result TC[L,R]
```

Example

Consider a distributed system with two nodes. Let R be the multilevel relation illustrated in figure 2. Let the constraints enforced be {(x1,x4), (x2,x4), (x3,x4)}. Assume that node 1 stores the pairs (x1,x2) and (x3,x4) while node 2 stores the pair (x2,x3). Also assume that node 2 is the designated one. A trace of MTCUP is given below.

Phase 1:

R1 = {(x1,x2), (x3,x4)}
R2 = {(x2,x3)}
S1 = {(x1,x2)
S2 = {(x2,x3)}

Phase 2:

S = {(x1,x2), (x2,x3)}
S+ = (x1,x2), (x2,x3), (x1,x3)}

7. CONCLUSION

The development of intelligent database systems has marked a milestone in technology because database management systems can be used not only to process stored data but also to deduce new data. In many instances, the new data is generated by expressing the request in the form of a recursive query. In this section we have discussed the need for recursive query processing in secure database management systems and also described an attempt in designing algorithms for recursive query processing in secure database management systems. We concentrated on a special type of recursive query, called the transitive closure query which is of practical as well as of theoretical significance.

The results obtained in this paper are the following:

(i) A formula for specifying the transitive closure queries for multilevel relations using the least fixed point operator,
(ii) An integrity rule to be enforced on the security constraints in order to prevent certain inference violations,
(iii) An algorithm to enforce the integrity rule,
(iv) Two algorithms for computing the transitive closure of a multilevel relation. We consider centralized as well as distributed relations.

Future work in this area includes experimentally evaluating the algorithms. The performance of these algorithms could be determined by varying the parameters such as the number of tuples or the number of security constraints. Such research will eventually lead towards the design and development of secure intelligent database systems.

ACKNOWLEDGEMENT

The work reported in this paper was sponsored by the Department of Navy (SPAWAR). I thank Reid Gerhart and Deborah Bodeau for their comments on this paper.

REFERENCES

[AGRA87] Agrawal, R., and Jagadish, H., "Direct Algorithms for Computing the Transitive Closure of Database Relations," International Conference on VLDB, Brighton, England, September 1987.

[AHO79] Aho, A., and Ullman, J., "Universality of Data Retrieval Languages," Proceedings of the 6th Annual Symposium on Principles of Programming Languages, January 1979, pp. 110-117.

[BANC86] Bancilhon, F., and Ramakrishnan, R. E., "An Amateur's Introduction to Recursive Query Processing Strategies," ACM-SIGMOD International Conference, Washington D.C., May 1986.

[BROD86] Brodie, M., and Mylopoulos, J., "On Knowledge Base Management Systems," Springer Verlag, 1986.

[CHAN82] Chandra, A., and Harel, D., "Structure and Complexity of Relational Queries," Journal of Computer and Systems Sciences, Vol. 25, #1, August 1982.

[CODD70] Codd, E. F., "A Relational Model of Data for Large Shared Data Banks," Communications of the ACM, Vol. 13, #6, June 1987, pp. 377-387.

[COHE89] Cohen, B., "Merging Expert Systems and Databases," AI-EXPERT, Vol. 2, #, 1989, pp. 22-31.

[DENN87] Denning, D. E. et al, "A Multilevel Relational Data Model," Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA, April 1987.

[GRAU82] Graubart, R., and Woodward, J., "A Preliminary Naval Surveillance DBMS Security Model", Proceedings of the 1982 IEEE Symposium on Security and Privacy, Oakland, CA, April 1982.

[RAMN88] Ramnarayan, R., and Lu, H., "A Data/Knowledge Base Management Testbed and Experimental Results on Data/Knowledge Base Query and Update Processing," Proceedings of the ACM SIGMOD Conference, Chicago, IL, June 1988.

[STAC89] Stachour, P., and Thuraisingham, M. B.,"Design of LDV - A Multilevel Secure Database Management System," Accepted for publication in IEEE Transaction on Knowledge and Data Engineering.

[TARS55] Tarski, A., "Lattice Theoretical Fixed Point Theorem and Its Applications," Pacific Journal of Mathematics, Vol. 5, #2, 1955, pp. 285-309.

[THUR87] Thuraisingham, M. B., "Security Checking in Relational Database Management Systems Augmented With Inference Engines," Computers and Security, Vol. 6, #6, December 1987.

[VALD88] Valduriez, P., and Khoshafican, S., "Parallel Evaluation of the Transitive Closure of a Database Relation," International Journal of Parallel Programming, Vol. 17, #1, 1988, pp. 19-42.