OBJECT-ORIENTED APPROACH TO INTERCONNECTING TRUSTED DATABASE MANAGEMENT SYSTEMS

Bhavani Thuraisingham and Harvey Rubinovitz

The MITRE Corporation, Bedford, MA 01730

ABSTRACT

This position paper describes an object-oriented approach to connecting trusted database management systems. In particular, an object model of the heterogeneous environment, the operation of the system, and a possible approach to simulating the environment using object-oriented technology are discussed.

1. INTRODUCTION

Within many organizations there are a number of computerized databases scattered across several sites. Efficient access to the information contained in these databases as well as sharing it has become an urgent need. As a result, the increasing number of heterogeneous database systems need to be interconnected. For many applications of heterogeneous database systems, secure interoperability is essential. Furthermore, for military applications, it is also important that such systems support multilevel user/data handling capability.

Recently, there have been some developments in homogeneous trusted distributed database management systems (see, for example, [THUR90, RUBI92a, RUBI92b]) and research is now beginning on interconnecting trusted database management systems in a heterogeneous and autonomous environment (see for example [THUR91a, THUR91b]).

In this paper we first review the security issues involved in interconnecting heterogeneous database systems, which we presented at the 1991 ACM Workshop on Data Management Systems, and then describe our research on a specific topic. In particular, we describe the use of object-oriented approach to interconnecting trusted database management systems. The organization of this paper is as follows. In section 2, we identify issues on interconnecting trusted database systems. In section 3, we describe an object-oriented model for interconnection. Issues on simulating the environment using the object-oriented simulation methodology is discussed in section 4. In our previous work on trusted distributed database management system, we have found that simulation studies have been cost effective. Therefore, we feel that simulation studies would help toward understanding the environment. Future considerations are given in section 5.

2. ISSUES IN INTERCONNECTION

In this section we discuss the issues involved in interconnecting heterogeneous database systems and then discuss the security impact for each issue. We will also discuss some of the additional security concerns in interconnecting heterogeneous systems.

(i) Semantic Heterogeneity and Schema Integration: Not all of the databases in a heterogeneous architecture are represented by the same data model. Therefore, the different conceptual schemas have to be integrated. In a multilevel environment, the constructs of one multilevel data model have to be transformed into those of another. During the translation process it should be ensured that a user who does not have access to a particular entity with respect to one data model must not have access to the same entity with respect to a different data model.

(ii) Role of object-oriented approach: Two aspects of the object-oriented approach are being investigated for a heterogeneous architecture. One is to develop a generic data model using this approach. The idea here is for the users to have a generic view of the entire system. The second use of the object-oriented approach is the design of the software components of the heterogeneous database system. With respect to multilevel security, rules for the secure interaction of the objects at different security levels need to be developed.

(iii) Transaction processing: Work is being directed toward integrating the various transaction processing mechanisms. At the same time transaction processing algorithms are being adapted to handle multilevel security. However, integration of these adapted algorithms to function in a heterogeneous environment need to be investigated.

(iv) Query processing and optimization: One of the research areas here is to develop a global cost model for query optimization. In a multilevel environment, the individual cost models will depend to a great extent on the security policy enforced and the storage mechanism used. The global cost model would then depend on the global security policy that is enforced.

(v) Standardization efforts: The current standardization efforts include work on Remote Data Access, SQL, Transaction Processing, Remote Procedure Call, and Application Program Interface. Security impact on these efforts need to be investigated.

(vi) Other security issues: In addition to the security impact on the various issues identified for heterogeneous database systems, there are some additional security concerns. These include the following:

(a) Different security policies: Each DBMS could enforce its own security policy for mandatory, as well as discretionary security. In addition, different authentication and integrity mechanisms may be used. The different policies have to be integrated in order to provide a global security policy.

(b) Different granularity of classification: Even if the relational data model is utilized at all nodes, the granularity of classification could be different. For example, one system could enforce classification at the tuple level while the other system could enforce classification at the element level. Such differences need to be handled at the global level.

(c) Different classifications of the same entity: An entity could be classified at the Secret level at one node and yet be classified at the TopSecret level at the second node. If this is the case, then the global policy should resolve such inconsistencies.

(d) Different semantics of classification levels: A classification level at one node could mean something entirely different at another node. Again the global security policy should resolve such inconsistencies.

(e) Different accreditation ranges: One node could handle the range from Unclassified to Secret while another node could handle the range Confidential to TopSecret. If a TopSecret user needs to access the Unclassified information handled by the first node, then it should be ensured that information from the TopSecret user is not transmitted covertly into the node handling the Unclassified data.

If nodes are autonomous, where they may choose to join or leave a federation when they want to, then there are additional problems that must be considered. For example, in an autonomous environment, priority is usually given to requests of local users over requests of federated users. Autonomy may have conflicts with security if these users are at different security levels. A more detailed discussion of theses conflicts is given in [THUR91a].

Much research needs to be done if solutions are to be provided for the secure interoperability of heterogeneous database systems. The problem becomes even more complex if the heterogeneity is with respect to more than one issue that we have identified. For example, the various systems could enforce not only different security policies, but also utilize different multilevel data models. We have conducted a preliminary investigation of some of these issues. For example, schema translation techniques are addressed in [THUR92]. Query processing and transaction management under a limited heterogeneity, where nodes handle different accreditation ranges, are discussed in [THUR91b]. In this paper we address issues on utilizing the object-oriented approach to interconnecting different trusted DBMSs. Our approach is discussed in section 3.

3. AN OBJECT MODEL OF THE HETEROGENEOUS ENVIRONMENT

In this section we describe the essential points of an object model which is used to model the environment under consideration.[1] In our model every entity is an object. That is an object could be a federation, a node, a database, or a DBMS. We group collections of objects with similar properties into classes. The classes form class hierarchies. We support inheritance and encapsulation. The properties of a class are specified by instance variables.

We will illustrate the environment with examples. Figure 1 represents the environment partially. The classes include FEDERATION and NODE. The instances of the FEDERATION class are the various federations. Each federation has the following instance variables: the federation-ID, the collection of nodes which form the federation, the federated schema, the federated security policy, and an administrator or group of administrators (if there is one). NODE class has nodes as its instances. Each node has the following instance variables: the node ID, node-name, federations (the federations to which the node belongs), the accreditation range (i.e., the range of security levels processed by the node), the database system (this includes the local policy, the schema, the DBMS, and the database), the administrator,

1 For a discussion on object-oriented data models we refer to [BANE87].

```
Class: FEDERATION

Instance Variables:
ID: Integer
Name: String
Node-list: List(NODE)
Schema: SCHEMA
Policy: POLICY
Administrator: PERSON
---------
Methods: - - -
```

```
Class: NODE

Instance Variables:
ID: Integer
Name: String
Federation: List(FEDERATION)
Accreditation range:
        Range(LABEL)
---------
Methods: - - -
```

Figure 1. Sample Classes

local users, and global users. Administrator and users are instances of the PERSON class with instance variables which include Person-ID, Name, Type of user, and Nodes.

Note that each node has a database system as an instance variable. This is an object and represents the local database system. This system consists of the multilevel database, the local trusted database management system, the local schema, and the local security policy. That is the database system object is a composite object. The specification of the component objects are yet to be defined. Note also that the local trusted database management systems may be relational systems or object-oriented systems. At a higher level of abstraction, we do not distinguish between these systems. That is, the interface to the database system object is uniform. The actual methods which implement the functions may be different for the various types of data models utilized.

We discuss the operation of the system with an example. This example is illustrated in figure 2. Suppose node A wants to join the federation F. Node A sends a message to the class FEDERATION with federation F as a parameter (msg 1 in figure 2). Node A may also give some other information as to what information it needs from others and the information it is willing to share. When federation F gets the message, the corresponding method gets executed. The federated policy may be examined to see if A can join the federation (msg 2 in figure 2 where it is assumed that the administrator C maintains the federated policy). It may send messages to the nodes already part of the federation to see if all of these nodes are willing for A to join the federation (msg 3 and msg 4 in figure 2). If all checks are satisfied, F sends a message to node A to join the federation and it includes A as part of the list of nodes which belong to it. That is, the value of the instance variable of F which specifies the nodes gets updated. Node A in turn updates the value of its instance variable for the federation.

As another example, suppose a user of node A wishes to access some data at node B. Let us assume that the request is via a federation. Then node A sends a message to the federation which in turn sends a message to node B. The federation object may perform certain security checks. Node B may check to see that this user is a valid federated user. It then performs some security checks and then sends a message to access the data via the database system instance variable. Note that if nodes A and B handle different

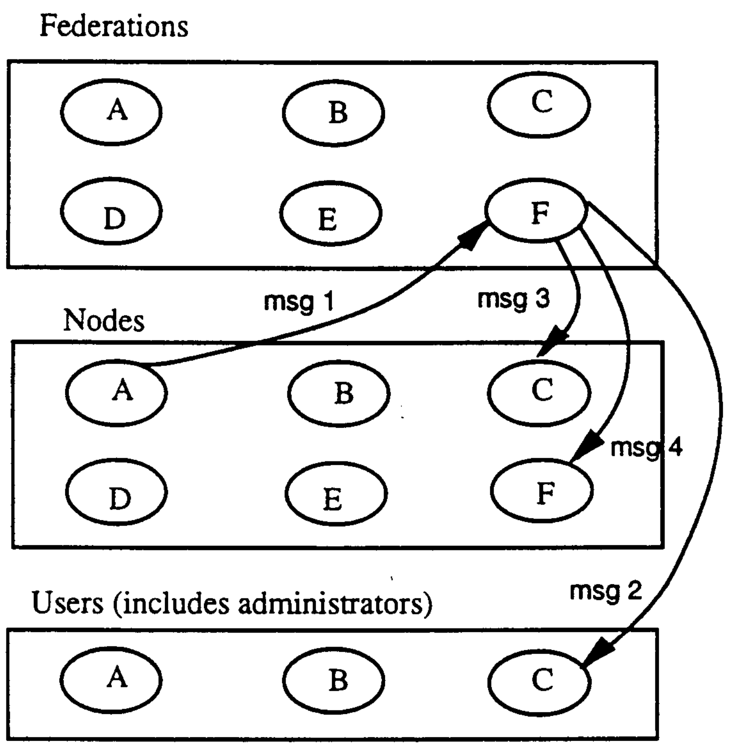

Figure 2. A Scenario

accreditation ranges then processing is more complex. For example, if the requesting user is TopSecret and Node B does not handle the TopSecret level, then the request has to be transmitted through a trusted process which must ensure that the request does not contain any potential TopSecret data. Further investigation is necessary before it is determined whether this trusted process is part of a method associated with the NODE class or it is part of the method associated with the FEDERATION class.

In this section we have stated just the essential points of the model and its operation. In reality we envisage that the environment will be rather complex. Therefore, we need to first develop the requirements for the environment (either hypothetical or possibly real) and then define the various classes and objects. Next, we need to develop techniques for all possible interactions between the various entities. Finally we need to show that the system is secure.

Before developing such a heterogeneous system it would be useful to simulate the environment to determine the performance of the various operations. Our previous work in trusted distributed database management systems has shown that simulation can be a valuable and cost effective tool [RUBI92b]. In section 4 we discuss issues on simulation. What is interesting about our approach is that we could use the object-oriented methodology for simulating our environment which is represented using an object-oriented model.

4. OBJECT-ORIENTED SIMULATION OF THE ENVIRONMENT

In this section we discuss how the environment described in section 3 could be simulated using an object-oriented simulation package.[2]

Before building an operational heterogeneous trusted distributed DBMS, it is essential that the system be simulated. This is because while prototyping has shown to be a valuable tool, due to resource constraints, it may be impossible for the prototype to depict the real world operational scenario. For example, the system may consist of thousands of physical nodes while the maximum number of nodes handled by prototypes are usually much less. Therefore, it is essential that simulation studies be carried out particularly for scenarios which cannot be handled by prototypes. Simulation models can give insights and quantitative results when a complete analytical approach would be impossible. Simulation has been considered a significant cost effective tool for modeling the behavior of distributed systems [RUBI89].

A trend in software development is that of object-oriented programming. Object-oriented programming techniques seem to be a natural match with simulation. Transferring a real world problem to a computer representation can be done with less effort and also allows for multiple levels of abstraction, incremental program development, and software reuse.

Using object-oriented techniques, a problem may be decomposed into a set of objects, where each object represents one object of the simulation model. An object is an encapsulation of data and code. The data refers to the current status of the object and the code describes the actions of the object.

The following code segments show how the examples presented in section 3 may be simulated. The code segments are patterned after MODSIM [MULL89] but due to limited space only the pseudo code is presented.

```
federation_object= OBJECT
        fed_id:                    INTEGER;
        set_of_nodes:              LIST;
        fed_schema:                SCHEMA;
        fed_security-policy:       POLICY;
        administrators:            LIST;

        TELL METHOD join(IN federation: federation);
        TELL METHOD remove_fed(IN federation: federation);

END OBJECT { federation_object }
```

2 Note that we have not yet carried out the simulation. We only discuss the design issues here.

```
OBJECT federation_object;
        TELL METHOD join(IN federation: federation);
        VAR
         allowed: boolean;
         BEGIN
         allowed:= TRUE;
         FOR EACH node IN set_of_nodes
         BEGIN
          allowed:=  send message asking if current node may join }
         END { for }
         IF allowed BEGIN
          add node to membership list
          return status ok
         ELSE
          return status false
        END  METHOD;
END OBJECT;

node_object= OBJECT
        node_id:                  INTEGER:
        node_name:                STRING;
        federation:               LIST of federation_object;
        accreditation_range:      LIST;
        db_policy:                POLICY;
        db_schema:                SCHEMA;
        db:                       DB;
        database:                 DATA;
        administrators:           PERSON;
        local_users:              PERSON;
        global_users:             PERSON;

        TELL METHOD valid_user(IN user_id: INTEGER);

END OBJECT { node_object };


OBJECT node_object;
        TELL METHOD valid_user(IN user_id: INTEGER);
        VAR valid_user:
        BEGIN
         valid_user:= check if user is a valid user of the federation checking with
                       accreditation ranges;
         return valid_user;
        END METHOD;
END OBJECT;
```

```
person_object= OBJECT
        person_id:              INTEGER;
        name:                   STRING;
        type_of_user:           USER;
        nodes:                  LIST of NODES;
END OBJECT { person }
```

4. SUMMARY AND FUTURE CONSIDERATIONS

In this paper we first provided an overview of interconnecting trusted database management systems and described an object-oriented model of the heterogeneous environment. Next we discussed issues in utilizing object-oriented simulation methodology for simulating such an environment.

Much remains to be done on this topic. First of all, we need to enhance the model that we have proposed. We have specified only the essential constructs of the model. Issues on handling autonomy and heterogeneity need to be investigated further and appropriate constructs need to be incorporated into the model. The object-oriented simulation methodology discussed here needs to be adjusted to fit the model. Simulation experiments need to be carried out for different topologies as well as for other parameters. Finally, the detailed design and implementation of the system needs to be carried out.

The recent trends in object-oriented approach to interconnecting heterogeneous database systems show much promise (see the discussion in [SHET90]). We believe that such a methodology would be useful for interconnecting trusted database management systems also.

ACKNOWLEDGEMENT: We thank Rae Burns and Arnon Rosenthal for discussions which contributed to this paper.

REFERENCES

[BANE87] [BANE87] Banerjee, J. et al., January 1987, "Data Model Issues for Object-oriented Applications," *ACM Transactions on Office Information Systems*, Vol. 5, #1.

[MULL89] Mullarney, A. et al., *MODSIM: a Language for Object-Oriented Simulation*, CACI, La Jolla, CA.

[RUBI89] Rubinovitz, H., 1989, *A Simulation Language for Distributed Databases*, Ph.D. Thesis, Department of Computer Science and Engineering, University of Connecticut, Storrs.

[RUBI92a] Rubinovitz, H. and B. Thuraisingham, "Design and Implementation of a Query Processor for a Trusted Distributed Database Management System, " To appear in the *Journal of Systems and Software*.

[RUBI92b] Rubinovitz, H. and B. Thuraisingham, June 1992, *Simulation of Query Processing and Concurrency Control, Algorithms for a Trusted Distributed Database Management System*, MITRE Technical Report, MTR 92B0000077 (a version also published in the proceedings of the 1991 and 1992 Computer Simulation Conferences).

[SHET90] Sheth, A., and J. Larson, September 1990, " Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases," *ACM Computing Surveys*, Vol. 22, #3.

[THUR90] Thuraisingham, B., July 1990, *Multilevel Security Issues for Distributed Database Management Systems*, Technical Report, MTP 291, The MITRE Corporation (a version also published in *Computers and Security Journal*, 1991).

[THUR91a] Thuraisingham, B., November 1991, *Security Issues for Federated Database Systems to Manage Distributed, Heterogeneous, and Autonomous Multilevel Databases*, Technical Report M91-78, The MITRE Corporation.

[THUR91b] Thuraisingham, B., and H. Rubinovitz, December 1991, *Design and Implementation Enhancements of the Prototype Secure Distributed Query Processor*, Technical Report M91-86, The MITRE Corporation (a version to appear in *Computers and Security* Journal).

[THUR92] Thuraisingham, B., 1992, "Schema Translation in Multilevel Heterogeneous Database Systems," to appear in *Database Programming and Design*.