International Journal of Engineering Research & Management Technology

May- 2015 Volume 2, Issue-3
ISSN: 2348-4039

Email: editor@ijermt.org www.ijermt.org

"Sharpening Skills..... Serving Nation"

Special Issue: 2nd International Conference on Advanced Developments in Engineering and Technology Held at Lord Krishna College of Engineering Ghaziabad, India

Security concerns for the Choice of Distributed Database Management System Model: Relational vs. Object-Oriented

Garima Bhardwaj, Priyanka Sharma
Assistant Professor
L.K.C.E, Ghaziabad

ABSTRACT
Object-Oriented Data Base Management Systems (OODBMS) and Relational Database Management Systems (RDBMS) share a common goal; to manage and protect mission critical data used by applications and individuals. In this regard it appears that any problem that may be solved by one database system may be solved by the other. This conception is not true. There are fundamental technical differences between OODBMS and RDBMS that make certain classes of problems tractable by OODBMS but not by RDBMS and vice versa. Productivity differences in application development between using an OODBMS or a RDBMS are sourced by these same fundamental technical dissimilarities. OODBMS possess certain features not provided by RDBMS that allow applications to offer benefits that would otherwise not be possible. The rapid growth of the networking and information-processing industries has led to the development of distributed database management system prototypes and commercial distributed database management systems (DDBMS). When choosing between the object-oriented model and the relational model for the development of distributed database, many factors should be considered. The most important of these factors are the various security concerns that includes single level and multilevel access controls, protection against data observation and inference, and maintenance of integrity When determining which distributed database model will be more secure for a particular application, the decision should not be made purely on the basis of available security features in both models, but the strength and efficiency of the delivery of these features should also be considered. Do the features provided by the database model provide adequate security for the proposed application? Does the implementation of the security controls add an unacceptable amount of computational overhead? This paper reviewed the security concerns of distributed databases. Also, the security strengths and weaknesses of both database models (relational and object oriented) are discussed.

Keywords: OODBMS, RDBMS, DDBMS,.

1. 1. INTRODUCTION
The rapid growth of the networking and information-processing industries has led to the development of distributed database management system prototypes and commercial distributed database management systems. A distributed system varies from a centralized system in one key respect: the database is stored in geographically separate sites which are inter-connected by some communication media. The aim of a distributed database management system (DDBMS) is to process and communicate data in an efficient and cost-effective manner. It has been recognized that such distributed systems are vital for the efficient processing required in military as well as commercial applications. Distributed database management

systems are subject to many security threats additional to those present in a centralized database management system (DBMS). For many of these applications it is especially important that the DDBMS should provide multi-level security. For example, the DDBMS should allow users who are cleared at different security levels access to the database consisting of data at a variety of sensitivity levels without compromising security.

For the past several years the most prevalent database model has been relational. Relational Database Management Systems (RDBMSs) have been very successful, but their success is limited to certain types of applications. As business users expand to newer types of applications, and grow older ones, their attempts to use RDBMS encounter the "Relational Wall," where RDBMS technology no longer provides the performance and functionality needed. This wall is encountered when extending information models to support relationships, new data types, extensible data types, and direct support of objects. Similarly, the wall appears when deploying in distributed environments with complex operations. ODBMSs offer a path beyond the wall. While the relational model has been particularly useful, its utility is reduced if the data does not fit into a relational table. Many organizations have data requirements that are more complex than can be handled with these data types. Relational databases typically treat complex data types as BLOBs (binary large objects). For many users, this is inefficient since BLOBs cannot be queried. In addition, database developers have had to contend with the hindrance mismatch between the third generation language (3GL) and structured query language (SQL). The hindrance mismatch occurs when the 3GL command set conflicts with SQL. There are two types of hindrance mismatches: (1) Data type inconsistency: A data type recognized by the relational database is not recognized by the 3GL. For example, most 3GLs don't have a data type for dates.(2) Data manipulation inconsistency: Most procedural languages read only one record at a time, while SQL reads records a set at a time. This problem is typically overcome by embedding SQL commands in the 3GL code. Solutions to both hindrance problems add complexity and overhead. Object-oriented databases have been developed in response to the problems listed above: They can fully integrate complex data types, and their use eliminates the hindrance mismatch. However, OODBMS has been created to address the growing complexity of the data stored in present database systems, but the development of adequate distributed database security has been complicated by this model. The security procedures in relational database model are more mature than the procedures in object oriented model. This is due to the fact that object-oriented databases are relatively new.

This paper will review the security concerns of distributed databases. Also, the security problems found in both models (relational and object oriented) will be discussed. Conclusively, we will compare the relative merits of each model with respect to security

.2. Background

2.1. Conventional Database Security aspects

Following are the requirements that must be satisfied by a secure single site database as well as a distributed database:

1. Physical Integrity (protection from data loss caused by power failures or natural disaster)
2. Logical integrity (protection of the logical structure of the database)
3. Availability
4. Data Accuracy
5. Access control (up to some degree depending on the data sensitivity)
6. Authentication
7. Protection of data from inference

Here, the requirements 4-7 are directly affected by the choice of database model. The principal goal of these requirements is to ensure data protection from unauthorized observation or inference, unauthorized modification, and from inaccurate updates. This can be accomplished by using access controls, concurrency controls, updates using the two-phase commit procedure (this avoids integrity problems resulting from physical failure of the database during a transaction), and inference reduction strategies (discussed in the next section). The level of access restriction depends on the sensitivity of the data and the degree to which the developer adheres to the principal of least privilege. A lattice is maintained in the DBMS that stores the access privileges of individual users. When a user logs on, the interface obtains the specific privileges for the user.
According to Pfleeger [Pflee89], access permission may be affirmed on the satisfaction of one or more of the following criteria: (1) Availability of data: Unavailability of data is caused by the locking of a particular data element by another subject, which forces the requesting subject to wait in a queue. (2) Acceptability of access: Only authorized users may view and or modify the data. In a single level system, it is relatively easy to implement. If the user is unauthorized, the operating system does not allow system access. On a multilevel system, access control is considerably more difficult to implement, because the DBMS must enforce the unrestricted access privileges of the user. (3) Assurance of authenticity: This includes the restriction of access to normal working hours to help ensure that the registered user is genuine. It also includes a usage analysis which is used to determine if the current use is consistent with the needs of the registered user, thereby reducing the probability of a fishing expedition or an inference attack.

Concurrency controls help to ensure the integrity of the data. It is the activity of coordinating concurrent accesses to a database in a multi-user database management system. These are particularly important in the effective management of a distributed system. Bell and Grisom [BellGris92] identify three possible sources of concurrency problems: (1) Lost update: A successful update was inadvertently erased by another user. (2) Unsynchronized transactions that violate integrity constraints. (3)Unrepeatable read: Data retrieved is inaccurate because it was obtained during an update. Each of these problems can be reduced or eliminated by implementing a suitable locking scheme (only one subject has access to a given entity for the duration of the lock) or a timestamp method (the subject with the earlier timestamp receives priority) [BellGris92].
Protection from inference is one of the unsolved problems in secure multilevel database design. Pfleeger[Pflee89] lists several inference protection strategies. These include data suppression, logging every move users make (in order to detect behavior that suggests an inference attack), and perturbation of data. As we will discuss later, the only practical strategy for the distributed environment that maintains data accuracy is suppression.

2.2. Security Aspects specific to Distributed Database Management Systems

2.2.1. Preserving data integrity in distributed database

According to Bell and Grisom [BellGris92], preservation of integrity is much more difficult in a heterogeneous distributed database than in a homogeneous one. The degree of central control dictates the level of difficulty with integrity constraints (enforcing the rules of the individual organization). The homogeneous distributed database has identical DBMS schema so can have strong central control. If the nodes in the distributed network are heterogeneous (the DBMS schema and the associated organizations are dissimilar), several problems can arise that will threaten the integrity of the distributed data. The listed problems are:
1. Inconsistencies between local integrity constraints,
2. Difficulties in specifying global integrity constraints,
3. Inconsistencies between local and global constraints [BellGris92].

Bell and Grisom described that local integrity constraints are bound to differ in a heterogeneous distributed database. The differences branches from differences in the individual organizations. Implementation of global

integrity constraints can eliminate conflicts between individual databases. Yet these are not always easy to implement.
Global integrity constraints on the other hand are separated from the individual organizations. It may not always be practical to change the organizational structure in order to make the distributed database consistent. Ultimately, this will lead to inconsistencies between local and global constraints. Conflict resolution depends on the level of central control. If there is strong global control, the global integrity constraints will take precedence. If central control is weak, local integrity constraints will.

2.2.2. Centralized or Decentralized Authorization

While developing a distributed database, one of the key points is to find is the system access point. Bell and Grisom [BellGris92] explained two strategies: (1) Users are granted system access at their home site. (2) Users are granted system access at the remote site.
The first case is easier to handle. It is similar to implement a centralized access strategy. Bell and Grisom point out that the success of this strategy depends on reliable communication between the different sites (the remote site must receive all of the necessary clearance information). Since, in distributed environment many different sites can grant access, the probability of unauthorized access increases. Once one site has been compromised, the entire system is compromised. Maintaining access control for all users at each site may reduce the impact of the compromise of a single site (provided that the intrusion is not the result of a stolen password).
The second strategy, while perhaps more secure, has several disadvantages. Probably the most glaring is the additional processing overhead required, particularly if the given operation requires the participation of several sites. Furthermore, the maintenance of duplicated clearance tables is computationally expensive and more prone to error. Also, the replication of passwords, even though they're encrypted, increases the risk of theft.
A third policy offered by Woo and Lam [WooLam92] centralizes the granting of access privileges at nodes called policy servers. These servers are arranged in a network. When a policy server receives a request for access, all members of the network determine whether to authorize the access of the user. Woo and Lam believes that separating the approval system from the application interface reduces the probability of compromise.

3. Security in Relational Database

3.1. Relational Database

The relational model is a database model that is based on first order predicate logic, first formulated and proposed in 1969 by Edgar F. Codd. The purpose of the relational model is to provide a declarative method for specifying data and queries: users directly state what information the database contains and what information they want from it, and let the database management system software take care of describing data structures for storing the data and retrieval procedures for answering queries. The fundamental assumption of the relational model is that all data is represented as mathematical n-ary **relations**, an n-ary relation being a subset of the Cartesian product of n domains. The relational model of data permits the database designer to create a consistent, logical representation of information. Consistency is achieved by including declared **constraints** in the database design, which is usually referred to as the logical schema. The theory includes a process of database normalization whereby a design with certain desirable properties can be selected from a set of logically equivalent alternatives. The access plans and other implementation and operation details are handled by the DBMS engine, and are not reflected in the logical model.

3.2. Security Aspects

3.2.1. Access Controls

The most common form of access control in a relational database is the view. The view is a logical table, which is created with the SQL VIEW command. This table contains data from the database obtained by SQL commands such as JOIN and SELECT. If the database is unclassified, the source for the view is the entire database. If, on the other hand, the database is subject to multilevel classification, then the source for the view is that subset of the database that is at or below the classification level of the user. Users can read or modify data in their view, but the view prohibits users from accessing data at a classification level above their own. In fact, if the view is properly designed, a user at a lower classification level will be unaware that data exists at a higher classification level [Denn87a]

In order to define what data can be included in a view source, all data in the database must be access classified. Denning [Denn87a] lists several potential access classes that can be applied. These include:
(1) Type dependent: based on the attribute associated with the data. (2) Value dependent: based on the value of the data. (3) Source level: Classification of the new data is set equivalent to the classification of the data source. (4) Source label: The data is arbitrarily given a classification by the source or by the user who enters the data. Classification of data and development of legal views become much more complex when the security goal includes the reduction of the threat of inference attacks. Inference is typically made from data at a lower classification level that has been derived from higher level data. The key to this relationship is the derivation rule, which is defined as the operation that creates the derived data (for example, a mathematical equation). A derivation rule also specifies the access class of the derived data. To reduce the potential for inference, however, the data elements that are inputs to the derivation must be examined to determine whether one or more of these elements are at the level of the derived data. If this is the case, no inference problem exists. If, however, all the elements are at a lower level than the derived data, then one or more of the derivation inputs must be promoted to a higher classification level [Denn87a].

The use of classification constraints to counter inference, beyond the protections provided by the view, requires additional computation. Thuraisingham and Ford [ThurFord95] discuss one way that constraint processing can be implemented. In their model, constraints are processed in three phases. Some constraints are processed during design (these may be updated later), others are processed when the database is queried to authorize access and counter inference, and many are processed during the update phase. Their strategy relies on two inference engines, one for query processing and one for update processing. According to Thuraisingham and Ford, the key to this strategy is the belief that most inferential attacks will occur as a result of summarizing a series of queries (for example, a statistical inference could be made by using a string of queries as a sample) or by interpreting the state change of certain variables after an update. The inference engine for updates dynamically revises the security constraints of the database as the security conditions of the organization change and as the security characteristics of the data stored in the database change. The inference engine for query processing evaluates each entity requested in the query, all the data released in a specific period that is at the security level of the current query, and relevant data available externally at the same security level. This is called the knowledge base. If the user's security level dominates the security levels of all of the potential inferences in knowledge base, the response is allowed [ThurFord95].

3.2.2. Data Integrity

The integrity constraints in the relational model can be divided into two categories: (1) implicit constraints and (2) explicit constraints. Implicit constraints which include domain, relational, and referential constraints enforce the rules of the relational model. Explicit constraints enforce the rules of the organization served by the DBMS. Pfleeger [Pflee89] lists several error detection methods, such as parity checks, that can be enforced by explicit constraints. Local integrity constraints are the examples of explicit constraints. Typically, explicit constraints are implemented using the SQL ASSERT or TRIGGER commands. ASSERT statements are used to prevent an integrity violation. Therefore, they are applied before an update. The TRIGGER acts as a response activation mechanism. If a problem with the existing database is detected (for example, an error is detected after a parity check), then a predefined action is initiated [BellGris92].

3.3. Security Aspects specific to Distributed Relational Database System

3.3.1. Access control using Global Views

As in the centralized relational database, access control in the distributed environment is accomplished with the view. In distributed database, view is developed from the global relations, instead of developing the view from local relations. So, it is referred to as a global view. The view mechanism is even more important in the distributed environment because the problem is typically more complex (more users and a more complex database) and while centralized databases may not be maintained as multilevel access systems, a distributed database is more likely to require the suppression of information [BellGris92].

Although global views are effective at data suppression and to a lesser extent at inference protection, their use can be computationally expensive. One of the key problems with a relational distributed database is the computation required to execute a complex query (particularly one with several JOINs, which join tables and table fragments that are stored at geographically separate locations).

3.3.2. Multilevel Constraint Processing in a Distributed Environment

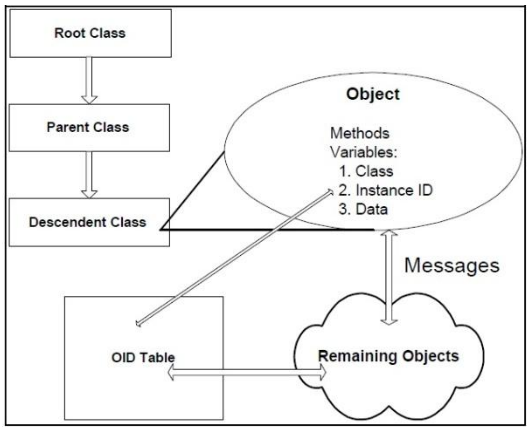
In an effort to provide additional inference protection beyond the global view, Thuraisingham and Ford extend their classification constraint processing model to the distributed environment. As with the centralized model, inference engines are added to the standard distributed database architecture at each site. Their model assumes that the distributed database is homogeneous. In this case, the inference engines at the user's site processes the query and update constraints. Only a small amount of overhead is added [ThurFord95]. If the distributed database is heterogeneous, however, then the processing overhead would be prohibitively expensive since the inference engines at each site involved in the action would need to process the security constraints for all the local data. Considering the processing demands already in place in a relational database management system (RDBMS), this appears to be impractical.

4. Object-oriented Database Security

4.1. Object-oriented Databases

An object-oriented database management system (OODBMS) is a database management system that supports the modeling and creation of data as objects. An object is defined by a class. This includes some kind of support for classes of objects and the inheritance of class properties and methods by subclasses and their objects.
An object is composed of two basic elements: variables and methods. An object holds three basic variables types: (1) Object class: This variable keeps a record of the parent class that defines the object. (2) Object ID (OID): A record of the specific object instance. The OID is also kept in an OID table. The OID table provides a map for finding and accessing data in the object-oriented database. (3) Data stores: These variables store data in much the same way that attributes store data in a relational tuple [MilLun92].

Methods are the actions that can be performed by the object and the actions that can be performed on the data stored in the object variables. Methods perform two basic functions: They communicate with other objects and they perform reads and updates on the data in the object. All control for access, modification, and integrity start at the object level.



Schematic view of the object-oriented model

4.2. Security Aspects

4.2.1. Access Controls

As with the relational model, access is controlled by classifying elements of the database. The basic element of this classification is the object. Access permission is granted if the user has sufficient security clearance to access the methods of an object. Millen and Lunt [MilLun92] describe a security model that effectively explains the access control concepts in the object-oriented model. Their model is based on six security properties:

Property 1 (Hierarchy Property): The level of an object must dominate that of its class object.
Property 2 (Subject Level Property): The security level of a subject dominates the level of the invoking subject and it also dominates the level of the home object.
Property 3 (Object Locality Property): A subject can execute methods or read or write variables only in its home object.
Property 4 (*-Property): A subject may write into its home object only if its security is equal to that of the object.
Property 5 (Return value property): A subject can send a return value to its invoking subject only if it is at the same security level as the invoking subject.
Property 6 (Object creation property): The security level of a newly-created object dominates the level of the subject that requested the creation [MilLun92].

Property 1 ensures that the object that inherits properties from its parent class has at least the same classification level as the parent class. If this were not enforced, then users could gain access to methods and data for which they do not have sufficient clearance. Property 2 ensures that the subject created by the receiving object has sufficient clearance to execute any action from that object. Hence, the classification level given to the subject must be equal to at least the highest level of the entities involved in the action. Property 3 enforces encapsulation. If a subject wants to access data in another object, a message must be sent to that object where a new subject will be created. Property 6 states that new objects must have at least as high a clearance level as the subject that creates the object. This property prevents the creation of a covert channel. Properties 4 and 5 are the key access controls in the model. Property 4 states that the subject must have sufficient clearance to update data in its home object. If the invoking subject does not have as high a classification as the called object's subject, an update is prohibited. Property 5 ensures that if the invoking subject from the calling object does not have sufficient clearance, the subject in the called object will not return a value. The object-oriented model and the relational model minimize the potential for inference in a similar manner. Remaining consistent with encapsulation, the classification constraints are executed as methods. If a potential inference problem exists, access to a particular object is prohibited [MilLun92].

4.2.2. Integrity

As with classification constraints, integrity constraints are also executed at the object level [MilLun92]. These constraints are similar to the explicit constraints used in the relational model. The difference is in execution. An object-oriented database maintains integrity before and after an update by executing constraint checking methods on the affected objects. As we saw in section 4.1.2., a relational DBMS takes a more global approach. One of the benefits of encapsulation is that subjects from remote objects do not have access to a called object's data. This is a real advantage that is not present in the relational DBMS. Herbert [Her94] notes that an object oriented system derives a significant benefit to database integrity from encapsulation. This benefit stems from modularity. Since the objects are encapsulated, an object can be changed without affecting the data in another object. So, the process that contaminated one element is less likely to affect another element of the database.

4.3. Security Aspects specific to Distributed Object Oriented Database System

Sudama [Sud95] states that there are many restrictions to the successful implementation of a distributed object-oriented database. The organization of the object-oriented DDBMS is more difficult than the relational DDBMS. In a relational DDBMS, the role of client and server is maintained. This makes the development of multilevel access controls easier. Since the roles of client and server are not well defined in the object-oriented model, control of system access and multilevel access is more difficult. System access control for the object-oriented DDBMS can be handled at the host site in a procedure similar to that described for the relational DDBMS. Since there is no clear definition of client and server, however, the use of replicated multisite approval would be impractical. Multilevel access control problems arise when developing effective and efficient authorization algorithms for subjects that need to send messages to multiple objects across several geographically separate locations. According to Sudama [Sud95], there are currently no universally accepted means for enforcing subject authorization in a pure object-oriented distributed environment. This means that, while individual members have developed their own authorization systems, there is no pure object-oriented vendor-independent standard which allows object-oriented database management systems (OODBMS) from different vendors (a heterogeneous distributed system) to communicate in a secure manner. Without subject authorization, the controls described in the previous section cannot be enforced. Since inheritance allows one object to inherit the properties of its parent, the database is easily compromised. So, without effective standards, there is no way to enforce multilevel classification. Sudama [Sud95] notes that one standard does exist, called OSF DCE (Open Software Foundation's Distributed Computing Environment), that is vendor-independent, but is not strictly an object-oriented database standard. While it does provide subject authorization, it treats the distributed object environment as a client/server environment as is done in the relational model. He points out that this problem may be corrected in the next release of the standard. The major integrity concern in a distributed environment that is not a concern in the centralized database is the distribution of individual objects. Recall that a RDBMS allows the fragmentation of tables across sites in the system. It is less desirable to allow the fragmentation of objects because this can violate encapsulation. For this reason, fragmentation should be explicitly prohibited with an integrity constraint [Her94].

5. Comparative study

We have seen that the choice of database model significantly affects the implementation of database system security. Each model has strengths and weaknesses. It is clear that more research has been completed for securing centralized databases. Sound security procedures exist for the centralized versions of both models. Both have procedures available that protect the secrecy, integrity, and availability of the database. For example, multilevel relational DBMS use views created at the system level to protect the data from unauthorized access.

OODBMS, on the other hand, protect multilevel data at the object level through subject authorization and limitation of access to the object's methods. The principle unsolved problem in centralized databases is inference. The current strategies do not prevent all forms of inference and those suggested by Thuraisingham and Ford are computationally cumbersome. Given that both models have well-developed security procedures, the choice of DBMS model in a centralized system could be made independent of the security issue. The same cannot be said of distributed databases. The relational model currently has a clear edge in maintaining security in the distributed environment. The main reason for the disparity between the two models is the relative immaturity of the distributed object-oriented database. The relational model, however is not without problems: The processing of global views in a heterogeneous environment takes too long, and the enforcement of database integrity in a heterogeneous environment is problematic because of the conflicts between local and global integrity constraints. The lack of completely compatible, vendor-independent standards for the distributed OODBMS relegates this model to a promised, yet not completely delivered, technology. If the distributed environment is homogeneous, the implementation of subject authorization should be possible. For the heterogeneous distributed OODBMS, however, the absence of universally accepted standards will continue to hamper security efforts.

6. Conclusion and Opportunities for Further Research

We have discussed database security issues in general and how the database model affects database system security in particular. We have seen that security protections for OODBMS and RDBMS are quite different. Each model has significant strengths and weaknesses. Currently, the RDBMS is the better choice for a distributed application. This is due to the relative maturity of the relational model and the existence of universally accepted standards. The recent emergences of hybrid models that combine the features of the two models discussed raise many new security questions. For example, Informix's Illustra combines a relational database schema with the capability to store and query complex data types. They call this system an "object-relational database." Informix claims that their system has all the capabilities of a RDBMS, including "standard security controls" with the principle advantage of an OODBMS: encapsulation, inheritance, and direct data access through the use of data IDs [Inf96].

This hybrid and similar systems offered by Oracle and others raise many new questions. For example, do the relational database security controls work well with complex data types and objects? How well do these security controls interface with encapsulation and object methods? What new avenues of attack have been opened by the combination of these two seemingly different concepts? What special security problems will arise when the object relational system is extended to the distributed environment?
In addition to the questions raised above, there are also opportunities for research in several other areas. They include subject authorization strategies for heterogeneous distributed systems, inference prevention strategies for both centralized and distributed database systems, and distributed object-oriented database security standards.

REFERENCES

1. [Bellgris92] Bell, David And Jane Grisom, Distributed Database Systems. Workinham, England: Addison Wesley,1992.
2. [Bert92] Bertino, Elisa, "Data Hiding And Security In Object-Oriented Databases," In Proceedings Eighth International Conference On Data Engineering, 338-347, February 1992.
3. [Denn87a] Denning, Dorothy E. Et Al., "Views For Multilevel Database Security," In Ieee Transactions On Software Engineering, Vse-13 N2, Pp. 129-139, February 1987.
4. [Denn87b] Denning, Dorothy. E. Et Al., "A Multilevel Relational Data Model". In Proceedings Ieee Symposium On Security And Privacy, Pp. 220-234,1987.
5. [Haig91] Haigh, J. T. Et Al., "The Ldv Secure Relational Dbms Model," In Database Security, Iv: Status And Prospects, S. Jajodia And C.E. Landwehr Eds., Pp. 265-269, North Holland: Elsevier, 1991.
6. [Her94] Herbert, Andrew, "Distributing Objects," In Distributed Open Systems, F.M.T. Brazier And D. Johansen Eds., Pp. 123-132, Los Alamitos: Ieee Computer Press, 1994.
7. [Inf96] "Illustra Object Relational Database Management System," Informix White Paper From The Illustra Document Database, 1996.
8. [Jajsan90] Jajodia, Sushil And Ravi Sandhu, "Polyinstantiation Integrity In Multilevel Relations," In Proceedings Ieee Symposium On Research In Security And Privacy, Pp. 104-115, 1990.
9. [Millun92] Millen, Jonathan K., Teresa F. Lunt, "Security For Object-Oriented Database Systems," In Proceedings
10. Ieee Symposium On Research In Security And Privacy, Pp. 260-272,1992.
11. [Mull94] Mullins, Craig S. "The Great Debate, Force-Fitting Objects Into A Relational Database Just Doesn't Work Well. The Impedance Problem Is At The Root Of The Incompatibilities." Byte, V19 N4, Pp. 85-96, April 1994.
12. [Sud95] Sudama, Ram, "Get Ready For Distributed Objects," Datamation, V41 N18, Pp. 67-71, October 1995.
13. [Thurford95] Thuraisingham, Bhavani And William Ford, "Security Constraint Processing In A Multilevel Secure
14. Distributed Database Management System," Ieee Transactions On Knowledge And Data Engineering, V7 N2, Pp. 274-293, April 1995.
15. [Woolam92] Woo, Thomas Y. C., And Simon S. Lam, "Authorization In Distributed Systems: A Formal Approach," In Proceedings 1992 Ieee Symposium On Research In Security And Privacy, Pp. 33-51,1992.