Information & Management 24 (1993) 93–103
North-Holland

Research

Multilevel security for information retrieval systems

Bhavani Thuraisingham

The MITRE Corporation, Bedford, MA, USA

In this paper, we describe multilevel security issues for information retrieval database management systems. We first discuss security issues for document representation; in particular, issues on developing an appropriate data model for representing multilevel information retrieval applications are given. Then we consider the security issues for document manipulation.

Keywords: Information retrieval database systems; Multilevel security; Document representation and manipulation; Data model; System architecture; Security constraints

Bhavani Thuraisingham is a lead engineer with the MITRE Corporation where she has initiated R & D activities in secure distributed database systems, secure object-oriented database systems, and secure intelligent database systems. She is currently working on multilevel object-oriented data models, approaches to handle the inference problem, theory of concurrency control for multilevel databases, and modeling the security engineering process. Previously she was at Honeywell Inc. where she was involved with the design of Lock Data Views, and before that at Contral Data Corporation where she was involved with the development of CDC-NET. She was also an adjunct professor of computer science and a member of the graduate faculty at the University of Minnesota. Dr. Thuraisingham received the M.S. degree in computer science from the University of Minnesota, M.Sc. degree in Mathematical Logic from the University of Bristol, and the Ph.D. degree in Recursive Functions and Computability theory from the University of Wales, Swansea. She has published over thirty journal articles in database security, distributed processing, artificial intelligence applications, and computability theory. She is a member of the Editorial Board of the Journal of Computer Security and serves as the program chair of the 6th IFIP 11.3 working conference in database security.

Correspondence to: Bhavani Thuraisingham, The MITRE Corporation, Bedford, MA01730, USA.

1. Introduction

Due to the explosion in demand for information, retrieval techniques have become indispensable to the efficient usage of computerized library facilities. These library facilities include the traditional information retrieval systems [VANR79] and the more recent hypermedia systems [ACM88]. However, easy access to information has also provided an avenue for malicious users to abuse these facilities which result in breaches in security. [1]

Current information systems for library use provide very little security. If a user has password access to an information system, then there is immediately access to almost all the information stored in that system. Recently, some hypermedia systems, such as HAM [CAMP88], have been developed which incorporate discretionary security measures. Here, access control lists are attached to the various entities stored in the database. Access control lists usually specify which user or group of users have read or write access to the entities. For many applications such as those used in the military environment, not all of the entities stored in the system have the same sensitivity levels. Furthermore, the users of these systems are also cleared to various security levels. That is, in such systems appropriate mandatory security controls are also needed. The intent is for the users to access and share a database with data at different sensitivity levels without violating security. Information retrieval database management systems that are capable of handling multilevel data and users are called Trusted In-

[1] We consider document processing systems, text processing systems, and information retrieval systems to be the same thing.

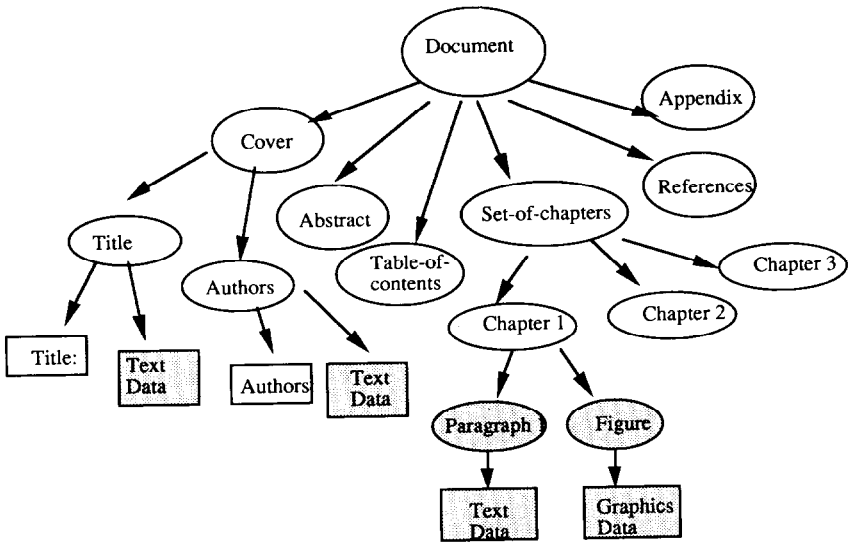

Fig. 1. Document representation.

formation Retrieval Database Management Systems (TIR-DBMS). [2,3]

In this paper, we investigate multilevel security issues for information retrieval database systems. We first describe the essential features of information retrieval systems ad then discuss the multilevel security impact on these systems. The security issues can be grouped into two categories. The first consists of issues on multilevel data representation and the second consists of issues on multilevel data manipulation.

[2] Note the following definitions. Discretionary security measures are usually in the form of rules which specify the type of access that users or groups of users may have to different kinds of data. In contrast, mandatory security controls restrict access to data depending on the sensitivity levels of the data and the clearance level of the user. By a sensitivity level, we mean the information attached to the data which reflects its sensitivity. Each user is cleared at a particular level which is called his clearance level. A multilevel database is a database in which the data are assigned different sensitivity levels. For a discussion of these definitions, we refer the reader to [TCSEC85].

[3] Such systems could also be called multilevel secure information retrieval database management systems.

2. Information retrieval system concepts

In this section, we discuss issues on document representation and manipulation.

2.1. Document representation

An initial step towards developing an information retrieval system is to develop a conceptual data model for representing documents. Such a model is essential, as users should not be burdened with the internal structure of the documents. A conceptual model for document representation is illustrated in *figure 1*, where a document is composed of cover, abstract, table-of-contents, set-of-chapters, references, and appendix. The cover consist of title and authors. Set-of-chapters consists of chapters 1, 2, and 3. Chapter 1 consists of a paragraph and a figure. The unshaded circles illustrate the standard portion of a document and the shaded circles illustrate "Authors:", etc.) ad shaded squares for nonstandard data (such as text, graphics, etc.).

Figure 2 illustrates the fact that two documents share the same paragraph. Document shar-

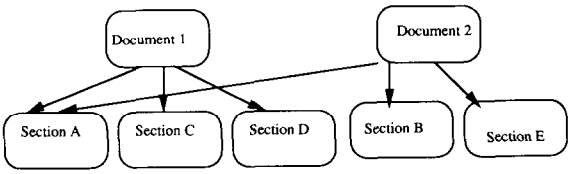

Fig. 2. Object sharing.

ing could reduce the storage overhead to a great extent. The data model should have the capability of representing the fact that different documents share their contents. Another important requirement that must be supported by the data model is versioning. For example, a document can undergo several iterations. Its evolvement over time should be represented by the model. Other desirable features that could be supported by the model include attribute specification and operation specification. Attributes describe properties of a document such as its author, publisher, and publication date. Documents are manipulated by the operations specified. For example, the operation "change-font-size", could be used to change the font size of a document.

In the beginning, simple graphical models were used to represent documents. More recently, semantic data models such as the entity relationship model and object-oriented data models have been used for this purpose. Object-Oriented data models (see, for example, WOEL86]) are becoming very popular for representing documents. For example, the IS-A and IS-PART-OF constructs enable the classification and representation of complex documents. The instance variable construct enables the specification of properties of documents. Through methods, the documents can be accessed and manipulated efficiently. Therefore, we will focus on the object model in discussing thc security issues.

2.2. Document manipulation

The document manipulation operations include browsing, on-line document querying, updating, and indexing. Browsing operation is involved with scanning a collection of documents or parts of documents starting from an initial point. Typically, a user would issue a list of key words. Objects called "key word objects" are scanned and the correct key word object is selected. From this object, the user can traverse an appropriate link to access the initial document. Browsing then continues by traversing links that connect the various documents.

There are essentially two ways to make use of the on-line query processor. One is a bibliographical search where a user would issue a list of key words and the system would then give out a list of publications. The second type of request is a document display where a user requests an entire document or parts of a document be printed. To retrieve a document or parts of a document, the user would have to use an appropriate document query language or write an application program. If the system is object-oriented, then query languages developed for the object-oriented data model could be utilized for this purpose. At present, research is also directed towards natural language-based query languages for information retrieval systems. [4]

When updating a document, multiple users may access it at the same time. Therefore, consistency of the document must be preserved. For example, if two users attempt to update a document at the same time or a user attempts to update a document while another user is reading it, then appropriate concurrency control techniques must ensure that the document is left in a consistent state.

Indexing is necessary to maintain large documents. It is often generated from the text of the document. The most popular index method used is the key word index. Here, the index objects contain key words that have pointers to the specific parts of a document. Indexing can be done automatically, where the index objects are updated by the system when the writer updates a document, or indexing is done declaratively where the writer makes indexing decisions. [5]

[4] Note that in [JAME85] three types of requests are described: bibliographical search, document assembly where a document is assembled and printed externally, and immediate display where a document is displayed on screen. We do not differentiate between the second and third requests.

[5] We have also carried out a preliminary investigation of security issues for multimedia system. This investigation is reported in [THUR90a]. Multimedia systems are more complex than document retrieval systems as they not only include textual and graphical data, but they also include animation, sound, and video data.

3. Security considerations

We now discuss the security impact on document representation and document manipulation.

3.1. Document representation

Since the object-oriented data model supports the data representation requirements imposed on documents at the conceptual level, the underlying secure data model is based on such a model. We consider the ORION data model which has been developed at MCC [BANE87] and discuss multilevel security issues for such a model. We state a set of security properties for the support of generalization/specialization hierarchy, attribute/method specification, complex objects, and versioning. [6]

The following security properties SP1 and SP2 are the elementary properties of the model.

Security Property SP1: If o is an object (either an object-instance, class, instance variable, or a method) then there is a security level L such that Level(o) = L.

Security Property SP2: All basic objects (example, integer, string, boolean, real, etc.) are classified at system-low.

The following property SP3 is required as it makes no sense to classify a document at the Secret level while the document class which describes the structure of a document is at the TopSecret level. On the other hand, a Secret document class could have Secret and Top Secret document instances. [7]

Security Property SP3: The security levels of the instances of a class dominate the security level of the class. (See *Figure 3.*)

The property SP4 is needed because there should not be an Unclassified English document if all documents are to be classified Secret.

[6] A more detailed discussion of the model is given in [THUR90b].

[7] Note that system-low (system-high) is the lowest (highest) level supported by the system. We assume that the security levels forms a lattice with Unclassified < Confidential < Secret < TopSecret.

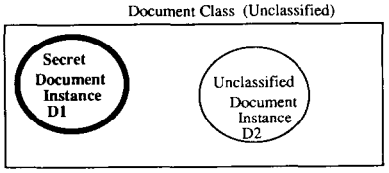

Fig. 3. Class-instance relationship.

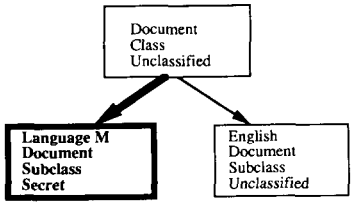

Fig. 4. Class-subclass relationship.

Security Property SP4: The security level of a subclass must dominate the security level of its superclass. (See *Figure 4.*)

Next, we describe alternate security properties that may be enforced on the instance variables (we consider simple and complex instance variables) and methods. Two ways to assign security levels to instance variables of a class are as follows:

Security Property SP5: The security level of an instance variable of a class is equal to the security level of the class. (See *Figure 5a.*)

Security Property SP5:* The security level of an instance variable of a class dominates the security level of the class. (See *Figure 5b.*)

If the property SP5 is enforced, the objects are assumed to be single level. If the property SP5* is enforced, the objects could be multilevel. 4t the conceptual level at least, the model should reflect the real-world closely. Two ways to assign security levels to methods are

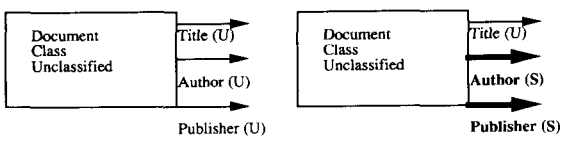

Fig. 5. Class-instance variable relationships.

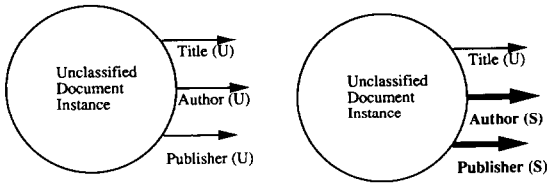

Fig. 6. Object-instance variable relationships.

Security Property SP6: The security level of a method of a class is equal to the security level of the class.

Security Property SP6:* The security level of a method of a class dominates the security level of the class.

Two ways to assign security levels to instance variables of an instance object of a class are

Security Property SP7: The security level of an instance variable of an object is equal to the security level of the object. (See *Figure 6a.*)

Security Property SP7:* The security level of an instance variable of an object dominates the security level of the object. (See *Figure 6b.*)

The next two security properties describe the relationships between the level of an instance variable and the level of its value.

Security Property SP8: The level of the value of an instance variable must be dominated by the level of that instance variable. (See *Figure 7.*)

Security Property SP9: If the instance variable c of an object is a complex instance variable, the security level of c is L, and if o1, o2,..., on are the objects that form the value of the instance variable c, then the security levels of o1, o2,..., on are dominated by L. (See *Figure 8.*)

The following are the security properties of versions of objects.

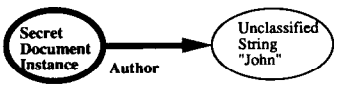

Fig. 7. Instance variable value relationship.

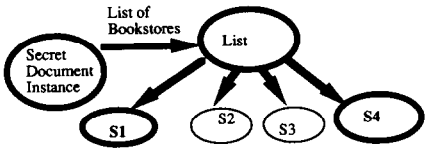

Fig. 8. Complex instance variable.

SP10: Let v be a version instance of the object o. Then Level(v) ≥ Level(o).

SP11: Let g be the generic instance of an object o. Then Level(g) = Level(o).

SP12: Let o′ have an instance variable link to version v of object o. Then Level(o′) ≥ Level(v).

SP13: Let o′ have an instance variable link to generic instance g of object o. Then Level(o′) ≥ Level(g).

SP14: Let o′ have an instance variable link to an object o. Let v′ be a version instance of o′. Then the instance variable link of v′ points to one of the following:

(i) NIL
(ii) o, provided Level(v′) ≥ Level(o)
(iii) generic instance g of o, provided Level(v′) ≥ Level(g), and
(iv) a version instance v of o, provided Level(v′) ≥ Level(v).

SP15: If c is an object (or a version) and c′ is a version of c and l is a version link from c to c′, then level(l) ≥ Level(c).

Figure 9 illustrates a version derivation hierarchy of: Unclassified object. Here, versions are

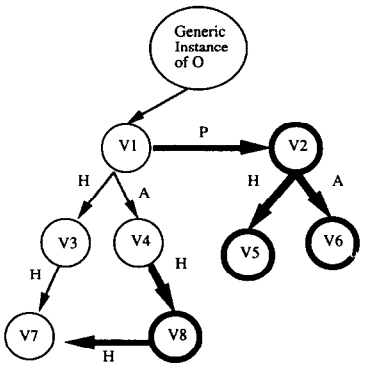

Fig. 9. Version derivation hierarchy.

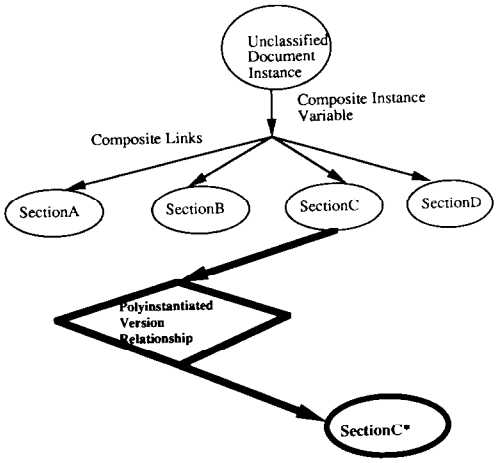

Fig. 10. Relationships between objects.

created within and across security levels. The link from an object (or version) to a version is called a version link. The generic instance has information on the version derivation hierarchy. Assuming that there are only two security levels, Unclassified and Secret, the generic instance stores Unclassified information of the hierarchy at the Unclassified level, and Secret information of the hierarchy at the Secret level. In Figure 9, the generic instance of object O has: Unclassified version instance V1. V2 is a polyinstantiated version of V1 at the Secret level. V3, V5, and V7 are historical versions of V1, V2, and V3, respectively. V4 and V6 are alternate versions of V3 and V5, respectively. V8 is a historical version of V4 at the Secret level. V7 is a historical version of V8. That is, over time, the Secret version V8 will evolve to the Unclassified version V7. We use P, H, and A for polyinstantiated, historical, and alternate versions respectively. Note that polyinstantiation is a mechanism to handle cover stories. It can be regarded as a form of versioning across security levels. [8] *Figure 10* illustrates polyinstantiation of sections.

[8] Versions are different interpretations of the same entity or event. Cover stories are usually false or inaccurate information that is generated at the lower security levels so that users at these lower levels do not deduce the sensitive information at the higher levels. Polyinstantiation is a mechanism that has been proposed in multilevel relational DBMSs to handle cover stories.

3.2. Document manipulation

We discuss some of the issues involved in designing a TIR-DBMS which supports the document manipulation requirements. In particular, security policy as well as architectural issues are considered.

3.2.1. Security policy

A basic mandatory security policy of an object-oriented database system is as follows: Subjects are the active entities, such as processes acting on behalf of users, and objects are the passive entities such as classes, instances, and methods. Subjects and objects are assigned security levels (we will use the terms subjects and users interchangeably). The following rules constitute the policy.

(i) A subject has read access to an object if the subject's security level dominates that of the object.
(ii) A subject has write access to an object if the subject's security is equal to that of the object. [9]
(iii) A subject can execute a method if the subject's security level dominates the security level of the method and that of the class with which the method is associated.
(iv) A method executes at the level of the subject who initiated the execution.
(v) During the execution of a method m1, if another method m2 has to be executed, then m2 can execute only if the execution level of m1 dominates the level of m2 and the class with which m2 is associated.
(vi) Reading and writing objects during method execution are governed by the properties (i) and (ii).

3.2.2. Architectural issues

Various architectures have been proposed for developing a Trusted DBMS. We believe that these architectures could be utilized to develop a TIR-DBMS also. They include (i) Operating System providing Mandatory Security, (ii) Kernelized, (iii) Integrity Lock, and (iv) Distributed Architecture [AFSB83].

[9] Note that we enforce the write-at-your-level policy and not the *-property (write-at-or-above-your-level) of the Bell and LaPadula policy [BELL75].

In the first architecture, the operating system provides the mandatory security. The TIR-DBMS basically runs as an untrusted application on top of the operating system. That is, the multilevel documents are decomposed and stored in single level fragments. The operating system controls access to the single-level fragments. In general, a subject should be able to read an object if the subject's security level dominates that of the object and a subject writes into an object if the subject's security level is that of the object. When a user asks for the contents of a document, the TIR-DBMS must construct the portion of the document from the fragments retrieved. In order to do this, the TIR-DBMS must maintain mappings between the conceptual representation of the multilevel document and its physical storage. In order to facilitate browsing a document, the TIR-DBMS must also maintain a network of nodes and links. Each node will map into one or more single-level physical storage objects. Users should then be able to browse through documents classified at or below their level. In this approach, the index files could be maintained on a per security level basis. That is, the index file for a document at level L is maintained at level L also. One of the major advantages of this architecture is that the TIR-DBMS need not be trusted. However, this would mean that the labels displayed on the documents cannot be trusted. Furthermore, the integrity of the data cannot be guaranteed.

In the second architecture, in addition to the operating system, the TIR-DBMS also enforces mandatory access control on its own objects. For example, consider a multilevel document consisting of Unclassified, Secret, and TopSecret sections. The document could be stored in a TopSecret segment. Access to this segment is controlled by the operating system. Then a trusted process which is part of the TIR-DBMS will control access to the individual portions of the document depending on the security level of the user. Index files could also be maintained by trusted processes. Such an approach is expected to give better performance over the first architecture. Furthermore, one could also trust the labels displayed in the document. However, the amount of trusted code that must be part of the TIR-DBMS needs to be determined.

In the third architecture, there are three major modules that make up the TIR-DBMS. An untrusted frontend, a trusted filter, and an untrusted back-end. Whenever data (which could be an entire document, a section, or even a paragraph) is entered, the trusted filter computes a checksum depending on the security level of the user and the value of the data. The data together with its checksum is stored by the untrusted back-end. When a user requests a retrieval operation, the back-end retrieves the data together with its label ad checksum and gives it to the filter. The filter recomputes the checksum for the data and label. If the new checksum does not match with the checksum. Since the data could be as large as a document or as small as a word, the checksum computation technique could be quite complex. Therefore, we believe that building a realistic system with this approach may not be feasible.

In the fourth architecture, a trusted front-end is connected to untrusted back-end machines, each operating at a single classification level. Several configurations have been proposed. On one end of the spectrum, there is no replication of data. That is, the machine operating at level L manages the database at level L. At the other end of the spectrum, there is total replication. That is, the data at level L is replicated at all databases at level $L^* > L$. The machine operating at level L' manages the database at level L'. In both architectures all communication is via the front-end. In the case of the replicated architecture, updates are straightforward as the user at level L updates the database at level L. Query processing becomes quite complex as the user's query may have to be decomposed and sent to the machines at or below the user's level. Also, if a request is sent to a lower level machine, then a check must be made that such a request does not contain sensitive information. For a multilevel document with classification at the paragraph level, the techniques would be quite complex. Browsing documents across different security levels also becomes quite complex. In the case of the replicated architecture, query processing is straightforward. Since data is replicated at the higher levels, a user's request at level L is only sent to the machine operating at level L. The update operation is more complex as the consistency of the replicated copies must be maintained. Furthermore, during updates the actions of a process at a higher level machine must not

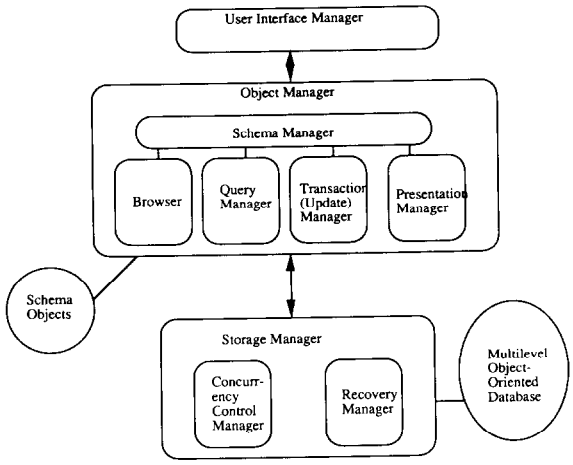

Fig. 11. System architecture for a TIR-DBMS.

interfere with a process at a lower level machine.

The architecture selected will depend on the requirements. For example, if a system has to provide high assurance, then one might favor the first architecture. If performance is a major concern, then the second architecture might be preferred.

3.2.3. A possible design

In this section, we discuss a possible design of a TIR-DBMS based on the first architecture. In this design, multilevel documents are stored in single-level fragments. The index files are also single level. Mandatory access to the single-level fragments is controlled by the operating system.

A system architecture for a TIR-DBMS based on the object model is shown in *Figure 11*. Users' requests are mediated by the User Interface Manager (UIM). The UIM is responsible for parsing the requests and generating an internal representation. It interfaces to the Object Manager (OM), which consists of five major modules: the Schema Manager (SM), the Browser, the Query Manager (QM), the Transaction Manager (TM), and the Presentation Manager (PM). The SM is responsible for manipulating the object-oriented representation of documents. The user's view of the multilevel database is object-oriented. The storage and manipulation of the structure of the documents is the responsibility of SM. The Browser is invoked when users ask to scan the various documents. The QM is responsible for query processing. That is, the user's queries on objects are processed by the QM and translated into an appropriate language so that the Storage Manager (STM) can execute the query. Similarly, the TM is responsible for managing the transactions on objects. The PM is responsible for ordering the documents so that they can be presented in a appropriate format to the user. Note that the Browser, QM, TM, ad PM all access the schema and structure information via the SM. The STM is responsible for storing and manipulating the data. It is the responsibility of the STM to decompose multilevel objects into single levels objects so that they ca be stored in single-level files or segments.

What is interesting about this architecture is that the STM, which is responsible for managing the storage objects, is separate from the OM, which represents and manipulates the objects at the conceptual level. Because of this separation, the STM need not be an object-oriented database system. That is, a relational database system could be used to manage the multilevel database. This way, one can take advantage of commercially available multilevel relational database systems for a near-term implementation of a TIR-DBMS.

It should be noted that, if the schema is stored at different security levels, then the access to these objects by the SM should be mediated by the operating system. For example, suppose that the existence of a paragraph of a document is Secret. When an Unclassified user requests for the structure of the document which contains the Secret paragraph, he should not be notified that it is not available. Thus the structure is entirely maintained by the SM component of the OM. Therefore, the access to the structure by the SM should be mediated by the operating system.

3.2.4. Security constraint processing

In a multilevel environment, each data entity (such as document, paragraph, section, etc) is assigned a security level. However, we have not yet described how the security levels may be assigned to these data entities. One option is for an object to be assigned the security level of its creator. However, if the multilevel world is complex and dynamic, then not only may several rules be used to determine the security level of an object but the level of the object may also change with time. Therefore, an effective tool is needed to classify and reclassify the objects as necessary.

In our approach, we use security constraints to assign security levels to the objects. They provide an effective and versatile classification policy. They can be used to assign security levels to the data depending on their content and the context in which the data is displayed. They can also be used to reclassify the data dynamically. We believe that processing security constraints is the first step toward controlling unauthorized inferences in a TIR-DBMS.

We have defined various types of security constraints. They include the following:

(i) Constraints that classify a database, a class, or an instance variable. These constraints are called simple constraints.
(ii) Constraints that classify any part of the database depending on the value of some data. These constraints are called content-based constraints.
(iii) Constraints that classify any part of the database depending on the occurrence of some real-world event. These constraints are called event-based constraints.
(iv) Constraints that classic associations between classes and instance variables. These constraints are called association-based constraints.
(v) Constraints that classify any part of the database depending on the information that has been previously released. These constraints are called release-based constraints.
(vi) Constraints that classify collections of instances. These constraints are called aggregate constraints.
(vii) Constraints which specify implications. These are called logical constraints.
(viii) Constraints that classify any part of the database depending on the security level of some data. These constraints are called level-based constraints.
(ix) Constraints which assign fuzy values to their classifications. These are called fuzzy constraints.

Our approach is to process certain security constraints during query processing, certain constraints during database updates, and certain constraints during database design. [10] When constraints are handled during query processing, they are treated as a form of derivation rules. That is, they are used to assign security levels to the data already in the database before it is released. When the security constraints are handled during update processing, they are treated as integrity rules. That is, they are constraints that must be satisfied by the data in the multilevel database. When the constraints are handled during database design, they must be satisfied by the database schema of a multilevel object-oriented database. [11]

One way to enforce security constraints in an object-oriented database system is to incorporate them as special types of methods. We call these constraint methods. The following alternate properties may be enforced:

C17: The security level of a constraint method is the security level of the class with which it is associated.

C17:* The security level of a constraint method dominates the security level of the class with which it is associated.

We also assume the previously discussed security policy for constraint method execution. A constraint method classified at level L can be executed by any subject classified at level L or higher.

[10] We treat security constraints as a form of integrity constraints enforced in multilevel database systems. The logic programming community [GALL78] has classified the integrity constraints into three groups. The first group consists of integrity rules that must he satisfied by the data in the database. The second group consists of derivation rules that are used to deduce new data from the data in the database. The third group consists of rules that are used to design the database. We have taken a similar approach for handling security constraints. That is, some constraints are handled during database updates, some during query processing, and some during database design.

[11] Note that, in a multilevel relational system, we specify security constraints as horn clauses [THUR87]. As a result, the techniques developed for checking the consistency as well as verifying the correctness of logic programs can be applied for validating the security constraints. However, in an object-oriented data model, we have used methods to specify constraints. Research is needed on techniques for checking the consistency of such constraints.

We illustrate constraint method execution with an example. Consider an Employee class with instance variables OID, SS#, Name, Salary, and Dept, and the following two constraints:

(T1) If Salary is greater than 50K, then an employee instance is Secret.
(T2) After 1/1/92, an employee instance is Secret if the Dept is Security.

Each constraint is specified as a constraint method. The first is a content-based constraint and we assume that it is processed during database updates. The second is an event-based constraint and it is processed during query operation. Specification of the schema is shown in *Table 1*.

Suppose an Unclassified subject requests the creation of an employee instance with instance variable values (10, John, 60K, Security). The constraint method for the content-based constraint is invoked. It is executed and the instance will be computed as Secret. It (or the calling method) must then specify whether to abort the update request or to create a Secret employee instance in a new session via a Secret subject.

Next, suppose an Unclassified subject requests the retrieval of all employee instances. The constraint method for the event constraint will be invoked during the-query operation. Fit is determined that the date is 3/1/92, then only the Unclassified employee instances whose department values are not 'Security' and have salaries less than or equal to 50K are returned to the querying subject.

Constraints that are processed during database design are the simple constraints, the association-based constraints. and the logical constraints. We illustrate with a simple example how such constraints may be enforced. Consider the following association-based constraint.

(T3) Name and Salary Instance Variables of the class EMP taken together are Secret.

Note that an Unclassified user could pose a query to retrieve the SS# and Name instance variable first and later pose a second query to retrieve the SS# and Salary instance variables. From the responses received for the two queries, the user could form the association between Names and Salaries. One solution to handle such a constraint would be as follows:

Table 1
Schema specifications.

```
CLASS DEFINITION:
   Name: EMP
   Level: Unclassified
Instance Variables:
   OID:    Class Integer,  Level Unclassified
   SS#:    Class Integer,  Level Unclassified
   Name:   Class String,   Level Unclassified
   Salary: Class Integer,  Level Unclassified
   Dept:   Class DEPT,     Level Unclassified
Update Methods:
UMethodI (EMP-Instance: Class EMP)
   If EMP-Instance.Salary > 50 K, then
      Level (EMP-Instance) = Secret
      {i.e. an instance variable of this instance is Secret}
   Endif
End UMethodI
Query Methods:
QMethodI (EMP-Instance: Class EMP)
   (If Date > 1/1/92, do the following)
   If EMP-Instance.Dept.Name = Security, then
      Level (EMP-Instance) = Secret
   Endif
End QMethodI
End Class Definition EMP
CLASS DEFINITION:
   Name: DEPT
   Level: Unclassified
Instance Variables:
   OID:    Class Integer, Level Unclassified
   D#.:    Class Integer, Level Unclassified
   DName:  Class String,  Level Unclassified
End Class Definition DEPT
```

Create three classes: EMP, EMP-ASSOC, and EMP-SAL. MP and EMP-SAL are Unclassified.

EMP-ASSOC is Secret. EMP has instance variables OID, SS#, Name, and Dept. EMP-SAL has instance variables OID and Salary. EMP-ASSOC has instance variables OID, SS#, Salary. Any instance of EMP-ASSOC must be at least Secret and have the association between SS#s and the salaries. Since the class definitions have changed, the constraints T1 and T2 may also need to be modified. Note that since the relationship be-

Table 2
Modified schema specifications.

CLASS DEFINITION:
Name: EMP
Level: Unclassified

Instance Variables:
OID: Class Integer, Level Unclassified
SS#: Class Integer, Level Unclassified
Name: Class String, Level Unclassified
Dept: Class DEPT, Level Unclassified

Query Methods:

QMethodI (EMP-Instance: Class EMP)
{If Date > 1/1/92, do the following}
If EMP-Instance.Dept.Name = Security, then
Level (EMP-Instance) = Secret
Endif
End QMethodI

End Class Definition EMP

{Class Definition for DEPT is as in table 2}

CLASS DEFINITION:
Name: EMP-SAL
Level: Unclassified

Instance Variables:
OID: Class Integer, Level Unclassified
Salary: Class Integer, Level Unclassified

End Class Definition EMP-SAL

CLASS DEFINITION:
Name:EMP-ASSOC
Level: Secret

Instance Variables:
OID: Class Integer, Level Secret
SS#: Class Integer, Level Secret
Salary: Class Integer, Level Secret

End Class Definition EMP-ASSOC

tween the salary instance variable and the other instance variables of the class EMP is no longer visible at the Unclassified level, the constraint T1 is no longer meaningful. The constraint T2 is still enforced on the class EMP. Specification of the revised schema is shown in *Table 2*.

4. Summary

The explosion in the quantity of documents that are being produced in almost all enterprises today has resulted in computerizing the library facilities. This has resulted in the development of sophisticated information retrieval systems. However, this also means that there is a greater chance of abuse of the information by untrusted users or the system. Many of the systems provide little or no form of security.

In this paper, we first stated the data representation and data manipulation requirements of information retrieval system applications and then discussed the security impact. in particular, we discussed issues on developing a multilevel data model for representing these applications and architectural issues for a TIR-DBMS. The discussion in this paper is the first step towards the design and development of a TIR-DBMS.

References

[ACM88] Communications of the ACM. July 1988, Special Issue on Hypertext Systems.

[AFSB83] Air Force Studies Board, 1983. Committee on Multilevel Data Management Security, *Multilevel Data Management Security*, National Academy Press.

[BANE87] Banerjee. J. et al., January 1987, "Data Model Issues for Object-oriented Applications," *ACM Transactions on Office Information Systems*, vol. 5, No. 1.

[BELL75] Bell, D. and L. LaPadula, 1975, *Secure Computer Systems: Unified Exposition and Multics Interpretation*, Technical Report No: ESD-TR-75-306, Hanscom Air Force Base, Bedford, MA.

[CAMP88] B. Campbell and J. Goodman. July 1988, "HAM: A general purpose hypertext abstract machine", Communications of the ACM.

[GALL78] Gallaire, H. and J. Minker. 1978, *Logic and Databases*, Plenum Press.

[JAME85] James, G., 1985, *Document Databases*, Van Nostrand, New York.

[TCSEC85] Trusted Computer Systems Evaluation Criteria, Department of Defense Document, 1985.

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

[THUR90a] Thuraisingham, B., September 1990, "Multilevel Security for Multimedia Systems," Proceedings of the 4th IFIP Database Security Workshop, Halifax. England.

[THUR90b] Thuraisingham, B., 1990, "Issues on Developing a Multilevel Secure Object-Oriented Data Model," MTP 394, The MITRE Corporation, Bedford. MA (a version published in the Journal of Object-Oriented Programming, Vol. 4, Nov/Dec 1991).

[VANR79] C.J. van Rijsbergen, "Information Retrieval", Butterworths, second edition, 1979.

[WOEL86] Woelk, D. et al., 1986, "Object-oriented Approach to Multimedia Databases," Proceedings of the ACM SIGMOD Conference.