Journal of International Information Management

Volume 6 | Issue 1 Article 3

1997

Security Versus Integrity in Information Systems

Gerhaed Steinke
Seattle Pacific University

Security Versus Integrity in Information Systems

Gerhard Steinke
Seattle Pacific University

1. Introduction

Security and integrity are frequently competing characteristics in an information system. Security implies that a user can only access a specific subset of the information in the system, namely that information which the user has permission to access. Integrity implies that the information is "correct", i.e., that it satisfies the constraints, rules and conditions contained in the information system. A problem arises when a user who is unable to access certain information because of security restrictions, is left with an "incorrect" or inconsistent view of the information system.

In this paper we define an information organizational structure and policy which permits security and integrity to co-exist. Our approach, called the xKB approach, specifies an area of the information system for those objects which meet the integrity requirements for a particular user but not the integrity constraints of the information system as a whole. Earlier versions and components of our approach are described in [Steinke, 1991].

Section 2 provides an example of the problem of providing security and maintaining integrity. Section 3 reviews past approaches to the problem and section 4 describes the xKB approach to solving the conflict between security and integrity. Section 5 provides a summary. Comments on the implementation of the xKB approach are found in section 6.

2. Description of Problem

Integrity means that the information system satisfies the constraints, rules and conditions which are contained in that subset, so that a user is provided with a "correct" subset of the information system. We use the term "correct" as defined by Meadows and Jajodia: An information system is correct if all data satisfies all known constraints [Meadows & Jajodia, 1988]. If each user has access to a "correct" subset of the information system, we say that multi-user integrity is satisfied.

In addition to multi-user integrity (i.e., the integrity of each subset of the information system), the integrity or correctness of the information system as a whole should also be a given. ll integrity constraints in the information system should be satisfied so that a correct information system is maintained, even if in practice (due to security requirements) no one may have access to all information.

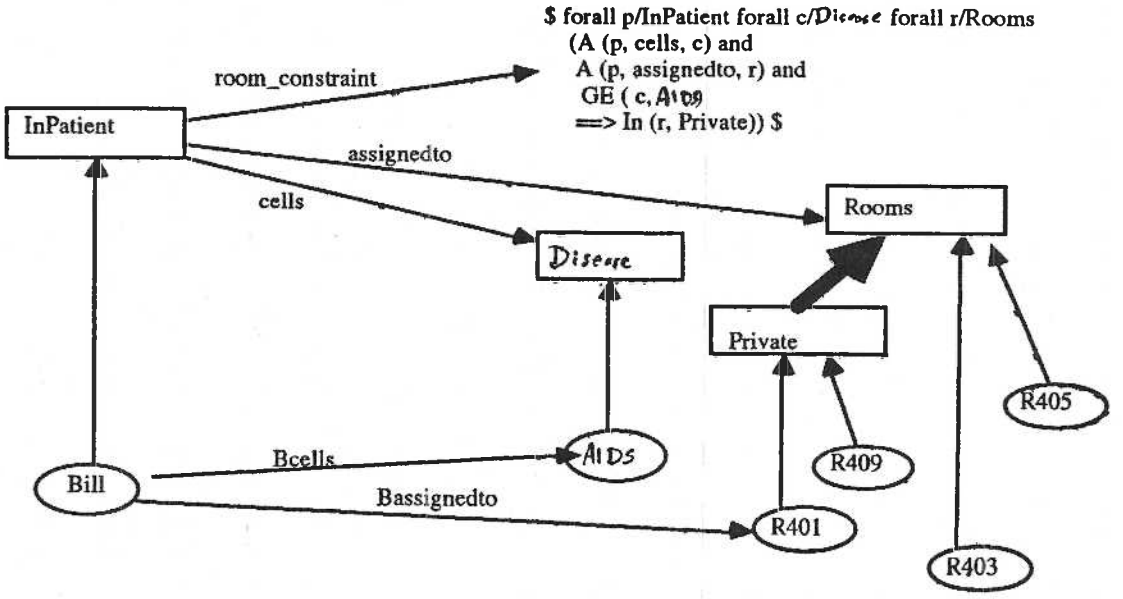
The problem of providing security and maintaining integrity (multi-user as well as the integrity of the whole information system) can be illustrated as follows: Normally modifications to an information system are only accepted if all constraints are satisfied. Constraints may contain information which should be hidden from some users. A user should not have to satisfy the constraints which he may not access. If a user has to satisfy hidden constraints, then he would discover their existence, and possibly the essence of the hidden constraints, thereby creating a breech of security. But if a user enters modifications which do not satisfy all integrity constraints, then the integrity of the information system as a whole is not maintained.

Constraints containing information which should be protected from some users are found in various applications and systems. In a personnel system a constraint may exist that an employee may not earn more than their boss. A clerk, who has the task to enter employee salary information is not given access to the salary constraint. If the clerk knew about the existence of such a constraint, the clerk could discover a manager's salary by trying different values to find the "limit" specified by the constraint. An environmental protection system may have the constraint which sends reports an environmental protection agency if a certain reading level is exceeded. If a company knows the readings that trigger a notification, they may be tempted to adjust their readings accordingly.

We will follow an example from a hospital information system which contains the following constraint: Patients with AIDS should be assigned to a private room. For various reasons hospital management does not want it to be known that such a constraint exists. In particular, a clerk, assigning rooms to patients should not know of the existence of the constraint. If the clerk did know of the constraint, then the clerk could try assigning a semi-private room to a patient to see if the assignment would be rejected. Thereby the clerk could infer whether a patient has AIDS or not.

A specific example with such a constraint is shown in fig. 1 (using the Telos knowledge representation standard [Mylopoulos, Borgida, Jarke & Koubarakis, 1990]. Private is a subclass of `Rooms`. **The class** `InPatient` has an attribute category `assignedto` which indicates a patient's room. The attribute category `Integer` provides a link to the reason for hospitalization. The room constraint specifies that if the count is greater than 200, i.e., disease is AIDS, then the person must be assigned to a room which is an instance of the class Private. In the example, Bill's room must be a private room, which it currently is. A user, with permission to make room assignments, who is not constrained by the room constraint could assign the room R403 to Bill. Yet this room assignment would not satisfy the integrity of the information system for those users who have access to the room constraint, Bill's room number and Bill's disease.

Figure 1. Example in graphical format



3. Past Approaches to Security and Integrity

During the last years numerous research efforts have focused on the problem of security and integrity. Past approaches to security and integrity tend to trade off, providing security for constraints on one side and requiring integrity on the other.

At one end of the spectrum, the problem is approached by **emphasizing integrity** and not allowing security criteria to be placed on constraints. If constraints are classified as public knowledge, then all users are aware of all constraints, and the integrity of the information system is maintained. Modifications which would cause the information system to lose its integrity are rejected, since all users know the constraints that have to be satisfied. In our example, the room constraint would be defined as public knowledge. Rooms can only be assigned if the room constraint is satisfied.

Meadows and Jajodia suggest that a way to compensate for making constraints public knowledge is by assigning higher security levels to the data which is related to the constraints [Meadows & Jajodia, 1988]. In such a structure the availability of information is reduced. In the room assignment example that would mean that while all users know that AIDS patients must be in private rooms, the fact whether a patient is

in a private room or not would receive a higher security level. For example, room assignments are no longer made by clerks but are entered by doctors.

Some protection may be assigned to constraints without affecting integrity by classifying constraints at the least upper bound of the security classes of the data over which they are defined [Meadows & Jajodia, 1988]. In other words, access to a constraint is not public, but is granted to those who have access to data which is related to the constraint.

Wiseman proposes an approach which depends on pre-specifying the security levels of those who may add or modify data which is related to constraints [Wiseman, 1990]. This approach does not consider the value of the information contained in the constraint itself, but forces a security level on the constraint which is related to the security of the associated objects. Users who may enter room assignments receive access to the constraint and therefore can only make modifications which satisfy the constraint. But not everyone knows of the constraint.

Another approach provides security for constraints by preventing a user from seeing the contents of constraints, while the system still checks all constraints before accepting modifications to the information system. Users receive "use" access for the constraint but are unable to read the content and makeup of the constraint. Modifications which do not satisfy the constraints are rejected, but without an explicit explanation. A user discovers the existence of a constraint, which is information in itself, but the constraint is protected. Although, with multiple attempts, a user may also be able to infer the essence of the constraint.

For some applications this latter option may be acceptable. In a life insurance application, all users know that a constraint exists to determine who will be accepted by an insurance company, even if they do not know the exact makeup of the constraint. In other situations, such as in the room assignment situation, it is essential that one is able to prevent a user from discovering the existence as well as the contents of the constraint.

The multilevel security model makes the claim that integrity constraints that relate values of data items stored at different access classes cannot be enforced since they are outside the mandatory security perimeter [Schell & Denning, 1986; Greenberg, 1991]. If the integrity checker could access information at a higher security level, then a Trojan Horse could also get in and leak higher level data. The question "whether there should be some notion of inter-level consistency, or how this might be specified, is unclear" [Schell & Denning, 1986:34].

There is also the concern that integrity constraints which are defined over data at several security levels may provide an inference channel so that low users may infer high information [Meadows & Jajodia, 1988]. By enforcing a notion of global consistency, one just opens up wide covert channels [Haigh, 1988].

If constraints are to be protected and **security is emphasized**, then integrity will suffer. The requirement that the information system as a whole must be correct is loosened. Only the constraints visible at a particular class or level are evaluated [Denning, Lunt, Schell, Shockley & Heckman, 1988]. This conflict

between information being entered and previously entered information is called the multi-party update problem [Keefe, Thomsen, Tsai & Hansch, 1989].

Polyinstantiation

An increasingly popular approach to the integrity problem is polyinstantiation. Polyinstantiation refers to the existence of duplicate data so that the integrity of each user's subset is maintained, although the integrity of the whole system is not considered.

In the room assignment example, polyinstantiation would mean that two rooms could exist for Bill. The one entered by the clerk which does not satisfy the integrity constraint and another one which satisfies the constraint. The latter room assignment is accessible to those who can access the constraint.

Polyinstantiation is used in the SeaView Security Model to achieve the goal that "database integrity holds with respect to the subset of the database visible at any security level." [Lunt et al., 1988:234].

There are benefits of polyinstantiation in terms of security which are not related to integrity. Polyinstantiation is able to implement cover stories to reduce inference [Lunt, 1991; Garvey & Lunt, 1991a]. A low user, with access to false information (cover story), is less likely to try to infer the existence of high information. Polyinstantiation also enables a high user to make a copy of low information which is frequently being modified, in order to achieve a stable copy of the information which the low user cannot modify [Garvey & Lunt, 1991].

But polyinstantiation comes with some serious problems. Polyinstantiation creates a problem for users with access to duplicate data. Which information is really correct? The user must somehow decide which information to believe and which to ignore. Similarly, the information system must choose which polyinstantiated information to use, e.g., in applying rules or in responding to queries. Duplicate or polyinstantiated information at different security levels does not necessarily imply that the value with the higher security level is the most accurate [Schell & Denning, 1986]. The more recent value may be the more accurate one. The higher your level, the more contradictory information is accessible [Wiseman, 1990].

Polyinstantiation also affects the operational semantics related to the updating of information [Lunt, 1990]. If multiple copies of information exists (at different security levels), then modifications made by a user may affect more than just the copy of the information which this user can access. These update semantics are discussed in [Lunt & Hsieh, 1991]. Constraints must also take into account polyinstantiated data to determine if the constraint is satisfied [Denning, 1988]. Laferriere suggests a predicate based approach to control and manage updates with polyinstantiated data [Laferriere, 1991].

Polyinstantiation fosters global inconsistency [Haigh, 1988]. With multiple versions of information, the goal of achieving an information system which satisfies its constraints is not possible.

Further Approaches to Integrity

It should be possible for users to add information without endangering the integrity of the base system [Summers & Kurzban, 1988]. Garvey and Lunt assign the responsibility for ensuring the consistency of information to individuals, namely the high user [Garvey & Lunt, 1991]. Berson and Lunt make the knowledge engineer (information system developer) responsible to define the constraints and rules so that the subset visible to even low level users make sense [Berson & Lunt, 1987].

In SeaView constraints do not have to be satisfied in all states, but only in the states that result from insert and update operations [Denning et al., 1988]. That leaves open the question what to do with a state in which the constraints are not satisfied.

Some systems have private databases, in addition to a common database [Dittrich, Hartig & Pfefferle, 1989]. The integrity problem is thereby amplified since consideration of the integrity of the private databases, the subset of the common database which a user may access, as well as the whole common database is involved. That reopens the question, whether integrity is indeed necessary.

Some say that integrity is an unreal requirement.While the real world may have constraints which always apply, the information system cannot satisfy all constraints due to judgment, a blur factor, incompleteness, etc. Integrity should not be required in a system with uncertainty or in a value based system, where information is not absolute, but is rather "believed" by certain users and therefore entered into the system, perhaps with a belief or trustworthiness factor.

The personal knowledge approach does not support integrity constraints, although a user may invoke a method which enforces or checks the integrity of his data [Biskup, 1990]. Integrity is not an issue in this approach.

Brodie says that systems would be "more realistic if they were able to knowingly support inconsistency and incompleteness" [Brodie, 1988:6]. Borgida presents an exception handling facility to handle the unusual, atypical, unexpected and exceptional cases [Borgida, 1985]. Exceptions to integrity constraints are to be accepted into an information system, not because of the security issue, but because exceptions to constraints occasionally are valid. Borgida proposes that these exceptions be marked and require "excuses" in order to provide accountability, as well as to inform and allow other users to navigate around the exceptions. Constraints must be able to coexist with information which conflicts with them.

The necessity for the satisfaction of constraints plays a role in semantic query optimization, since integrity constraints are used for translating a query into a form that may be answered more efficiently than the original [Chakravarthy, Grant & Minker, 1990]. If constraints are not all satisfied then query optimization cannot rely on constraints. Access structures and optimization are also dependent on integrity constraints [Biskup, 1990].

The issue whether constraints are consistent and finitely satisfiable is examined by [Bry & Manthey 1986; Bry, Decker & Manthey, 1988]. Morgenstern proposes a definition of sphere of inference of information, and constraints in particular, to enable one to know how far the impact of a constraint goes [Morgenstern, 1987].

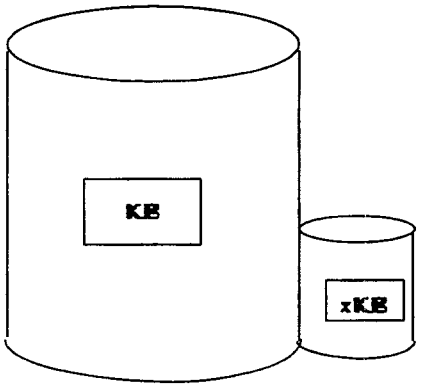
4. The xKB Approach

The goal of the xKB approach is to allow security to be placed on any information while at the same time maintaining integrity. The maintenance of integrity concerns the information system which each user may access, i.e., multi-user, as well as the whole information system. Not only should the subset of the information system which a user may access be correct, the information system as a whole should also satisfy all specified constraints. Although there may not be many users who may access the whole information system, it is still essential to have integrity as a characteristic of the whole information system.

The xKB approach has two components. The first component maintains the integrity of the whole information system. The second component enables each user to have a correct subset of the information system.

The first component of the xKB approach is the definition of an area which is added to the information system to contain that information which is acceptable for a particular user but does not satisfy all integrity constraints in the information system. The information system is divided into two sections called KB and xKB (fig. 2). The whole information system, by which we mean the information which meets all integrity requirements, is found in the KB. The xKB, on the other hand, is used to store objects created by users which do not meet the integrity requirements of the whole information system. The information in the xKB is not considered to be a part of the "real" information system. Objects which a user has permission to access, both in the KB and in the xKB, make up the users's information system.

Figure 2. Two components of the information system

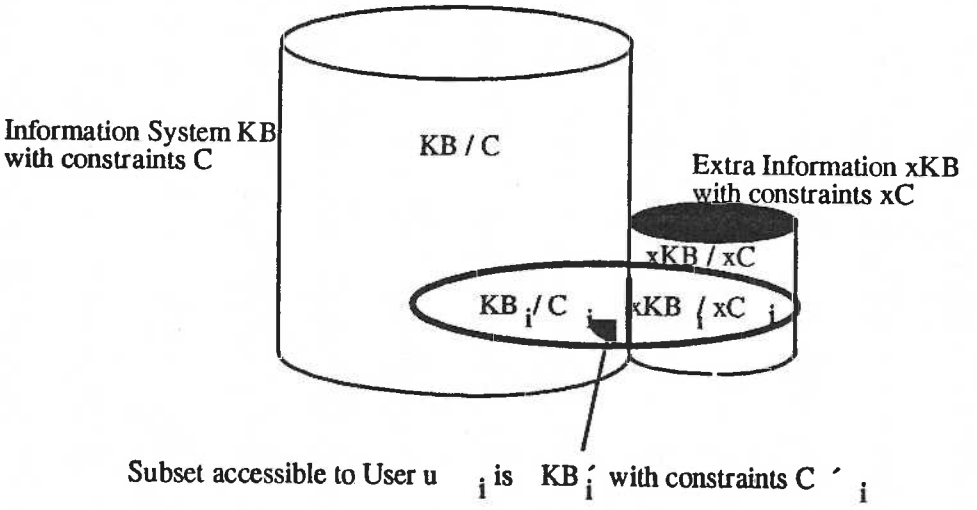


The second component of the xKB approach is the capability for a user to temporarily ignore objects which do not satisfy the integrity of his subset of the information system. A user may ignore objects by assigning them a pending status. For example, a user may ignore a constraint and thereby have a correct subset of the information system--although with operating restrictions.

4.1 xKB

More formally, the first component of the xKB approach can be described as follows (fig. 3).

Figure 3. Components of KB and xKB



The full information system which satisfies all constraints is KB/C. The information and the constraints which do not necessarily satisfy all constraints are in xKB. An individual user may have access to information in KB as well as xKB.

Modifying the Information System

A user makes a modification to the information system. Three aspects of integrity could be affected by the modification:

- the integrity of the user's subset of the information system,
- the integrity of the full information system,
- the integrity of another user's subset of the information system.

Integrity of Creator's Subset of the Information System

A modification to the information system must satisfy the integrity constraints which the user may access. If the modification does not meet the integrity constraints which are accessible to the user, the

modification is rejected. Since the user has access to the constraint which is not satisfied, the user can be told why the modification could not be accepted. This can be represented as follows:

Fig. 4 shows an algorithm which describes the integrity checking which is required in order to accept new information into the information system and how to determine where to place the new information, in the KB or in the xKB part of the information system.

The first decision concerns the integrity of the creator's subset of the information system. If the integrity requirements of this user's information system are not satisfied then the new information is rejected and the user is told why, since it concerns constraints which the user may access. If the integrity requirements of this user's information system are satisfied, then the new information can be accepted. As far as this user is concerned, the new information is valid.

Integrity of the Full Information System

A modification of the information system may satisfy the integrity of the creator's subset of the information system, but not satisfy the integrity requirements if all constraints, including the ones which this user may not access, are taken into consideration. Yet the modification should still be accepted. If the modification is not accepted then the user would discover the existence of a constraint which he cannot access.

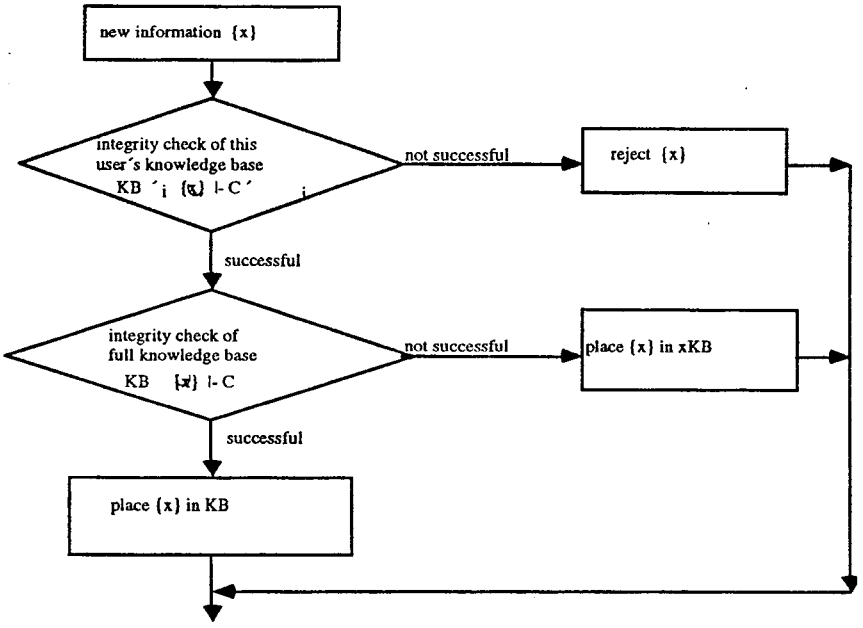
The xKB approach places an object which does not satisfy the integrity requirements of the full information system in the xKB portion of the information system. So the issue to be decided is where to place a modification in the information system, in the KB part or in the xKB part.

Therefore a second integrity check is carried out to determine if the new information maintains integrity in the whole information system, KB. If so, then the new information can be added to the KB. If not, then the new information is placed in the xKB, and not considered a part of the "real" information system. Thereby the integrity of the KB is maintained. But regardless where the information is placed, for this user the information that has just been added is an integral part of the information system which this user (and possibly other users) may access.

To the user this second check is transparent since the user should not be aware where the information is actually stored. If a user did discover where the information is stored, then the user would know whether his new information did, or did not meet all integrity constraints.

Figure 4. Integrity checking algorithm when considering new information

If the statement is true, then {x} can be added to KB (more specifically to KB_i' since as creator of the {x}, the user will have access to it). Otherwise {x} must be added to xKB (more specifically to xKB_i').



When modifications to the information system are considered, one first thinks of additions to the information system. Adding objects to the xKB does not affect the KB which always satisfies its integrity constraints. The situation could occur that a user may, according to the constraints which he has access to, be able to delete an object, but such a deletion would cause the whole information system to lose its integrity. If deletion is seen as the removal of an object in the KB, then the KB would lose its integrity. In an information system with a temporal component, deletion is the addition of an object which indicates that the referenced object is deleted. Since this additional object can be placed in the xKB, the KB still considers the "deleted" object as valid, whereas for the user who also considers the information contained in the xKB, the "deleted" object is indeed deleted.

Example

The room assignment example (fig. 1) is used to illustrate the first component of the xKB approach. UserA has permission to assign rooms and attempts to assign room R409 to InPatient Bill. he link to room R409 would be placed in the KB, regardless whether userA has access to the constraint or not, since the

information is valid, both when considering the whole information system and when considering the subset of the information system which userA sees.

If userA has access to the assignment constraint, and attempts to assign Bill to room R403 then the assignment would be rejected. UserA may know why since he has access to the constraint and therefore is aware of the constraint.

If userA does not have access permission to the constraint and enters room R403 for Bill, then this new information would be placed in the xKB (fig. 5). Since userA does not know of the assignment constraint, from userA's point of view, the information system is valid, so the new information is accepted. But this new information does not satisfy the integrity of the whole information system, KB, so it is placed in the xKB. The integrity of KB is maintained. The objects encircled in fig. 5 show the subset of the information system which user A may access.

But the acceptance of room R403 could also affect the integrity of the information system from the point of view of other users. This case is considered in the second component of the xKB approach.

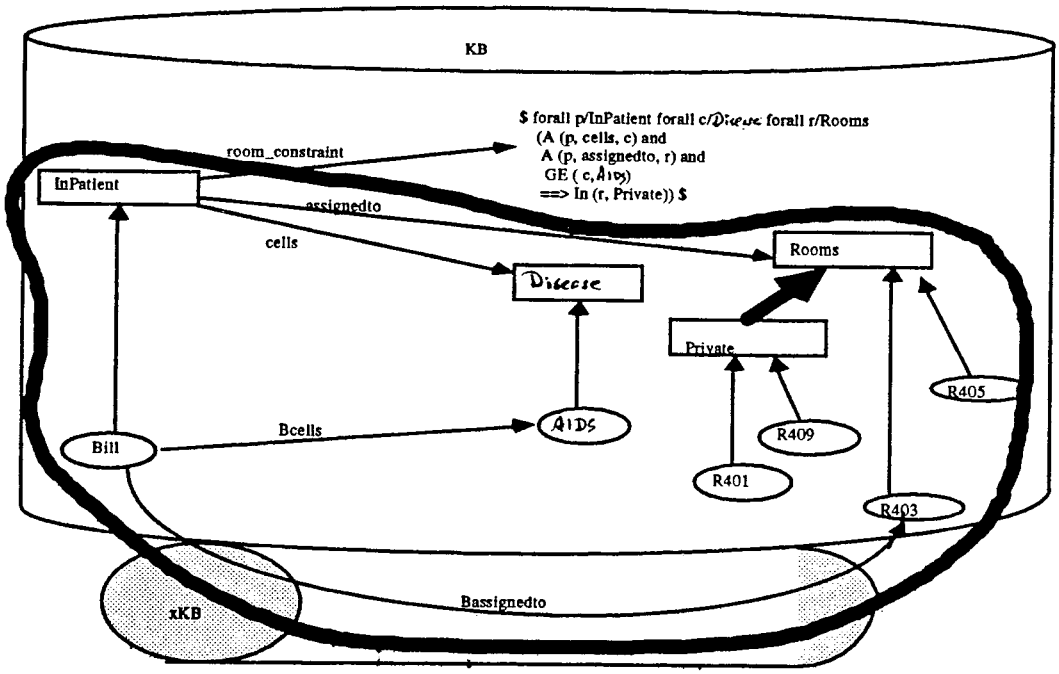
Divergent Information Systems

As more and more objects are placed into the xKB, because they do not meet the constraints in the whole information system, the concern grows, that two "divergent" information systems are being created. One information system consists of the information in the KB. The other information system consists of the additional information contained in the xKB, which does not satisfy all integrity requirements and is not considered to be a part of the "real" information system, but is referenced by individual users.

The larger the xKB becomes, the more information is being referenced by users which is not considered to be a part of the "real" information system. The purpose of the xKB is to accept information, even though the integrity of the whole information system is not satisfied, in order to prevent a user from discovering the content of constraints to which no access permission is granted. In order to prevent increasingly divergent information systems, there must be an active process of reconciling objects in the xKB with the KB, and thereby moving them to the KB.

An object in the xKB is treated just like an object in the KB, in that it may be made available to other users by the owner of the object. A user may access an object in the xKB in a similar manner as an object in the KB, only when the user receives permission to access the object. For some users, this new object will cause no problems, perhaps because they are also unable to access the constraint which this object does not satisfy. For those users who receive access to an object in the xKB and also have access to the constraint which this object does not meet, their information system no longer satisfies its integrity constraints. What should this user do?

Figure 5. Information in KB and xKB



A user who has access to objects in the xKB, and the constraints that are not satisfied, may have the capability to make modifications to the information system to remove the cause for the integrity failure. In addition, the modification may allow that one or more objects in the xKB can be moved to the KB. Since users are unaware of whether objects exist in the KB or the xKB, such movement is not noticeable to users. Rather, because a user has removed a problem of which he was aware, the system can transfer objects from the xKB to the KB.

In the example with the room assignment constraint, a user who has access to the constraint and the room assignment which does not satisfy the constraint, may have the authority to make the necessary adjustment to the room assignment. Thereby the room assignment existing in the xKB would be moved to the KB. If the user decided to delete the room constraint, then the room assignment could also be moved from the xKB to the KB.

If the user does not have the capability to make modifications to satisfy the integrity failure, there must be another way to achieve a correct subset of the information system. This is possible via the pending option discussed below.

4.2 Pending Option

The addition of an object, {x}, may affect the integrity of subsets of the information system of other users, since {x} may also be a part of the information system accessible to another user (regardless whether the object is in the xKB or the KB). The difficulty in solving this problem lies in the fact that the user making the modification does not, and should not, know if the integrity of another user is affected. If a user could tell what causes another user's information system to lose its integrity, then the former user would be able to infer information about the latter user, such as what objects the user may or may not access. Security requires that the information accessible by one user is not discovered by another user.

Integrity checking takes place when new information is added to the information system but not when access permission to objects is provided by one user to another user.

The problem of one user affecting the integrity of another user's subset of the information system is not just caused by constraints. In the discretionary security approach one user may grant another user permission to access some object. Integrity would be lost by granting access to incomplete information which does not satisfy the syntax requirements of the information system, e.g., a user may provide another user access to a characteristic of a patient, without providing access permission to the patient object itself. In the mandatory approach, a user who adds information at a low security level is thereby providing information which a user at a higher security level may also access. This new information may cause problems to the user at a higher security level.

Similarly the deletion of an object may affect another user. The effect of such action on another user's integrity is not of real concern to the owner of the information who deletes the information or revokes access permission.

The multiple situations which enable one user to affect the integrity of a user's subset of the information system makes it difficult to solve the problem completely. But the problem can be solved partially by allowing a user to temporarily ignore such objects which cause the failure of his integrity constraints, or ignore the integrity constraints themselves. A way to temporarily ignore objects is by marking one's access permission to these object with a pending status. A pending status means that the object is not considered to be a part of the user's subset of the information system. Marking an object as pending only affects the subset of the information system of the user who has marked the object. When the user cancels the pending status of an object then the object will again be considered to be a part of his information system.

This option to place information in a pending state corresponds to practice in the real world. A user may decide not to consider new information when it is received, but rather to examine it at some future time, perhaps when related information is available.

By placing objects in a pending state, namely those objects which do not satisfy the integrity constraints or the integrity constraint objects themselves, a user can achieve a correct information system. (The term correct is used to indicate that the information system satisfies all integrity constraints, although from a practical point of view, data is missing, namely the objects which have been marked as pending.) This

would allow a user to continue to work, knowing that at some time in the future or for particular operations, the pending objects have to be considered again.

Users are also reminded at the beginning of each session of the objects with a pending status. This reminder provides users with an opportunity to make modifications to the information system in order to remove the reason for the pending status.

Restrictions must be specified on the operations a user may perform when objects exist with a pending status. Otherwise, e.g., marking an integrity constraint as pending would allow one to add invalid data.

Discovering Loss of Integrity

Integrity is checked when information is added to the system, but not when information is added to the subset of the information system accessible to other users. If this was done, then every modification to the information system would require integrity checks for all users who have access to that modification. Besides increasing the time for integrity checking, a user whose change impacts the integrity of another user's information system, should not know of the problem, and therefore he could take no action. Action must be taken by each user for his own subset of the information system.

One must decide when to check the integrity of a user's information system. Greenberg suggests there should be a process which frequently probes data and re-establishes correctness as soon as possible [Greenberg, 1991]. This is expensive and constraints will often be violated again.

Our suggestion is to check the integrity of a user's information system when the user begins a session which accesses the information system. But a user's subset of the information system could lose its integrity, even while the user is working on it. For example, while one is working, another user provides access to an object which causes the recipient's information system to lose its integrity.

Working with Incorrect Information

This leads to the question whether it is indeed necessary for a user's subset of the information system to meet all integrity constraints. Could not a user be allowed to access the information system even though some integrity constraints are not satisfied? What if the user does not reference objects related to the inconsistency? Does further work on an incorrect information system only make the situation worse?

Considering the room assignment constraint example again, what effect does an incorrect assignment have? If during the current session the user does not reference patients and their rooms, then the presence of an "invalid" room has little effect--it really is not essential that the information system satisfies the room constraint during this session. Even if the user would access patient information, but not their rooms, then it would seem that such work could continue.

The problem is that at the start of a session the system does not know what objects will be accessed during the session. According to Borgida, users should receive warning when exceptional or incorrect values are being used [Borgida, 1985].

A concern is that the user who added the "invalid" information (which was placed in the xKB) will discover that the information has been changed by another user, and perhaps thereby conclude that a constraint exists. Nevertheless, if an error exists, than it must be corrected. If this correction should take place automatically or if the error should not even have been accepted in the first place, then the user would have been given access to the constraint.

5. Implementation of the Group Security Model in ConceptBase

We have implemented a prototype of the Group Security Model (GSM) in the information system management system ConceptBase [Jarke, 1991].

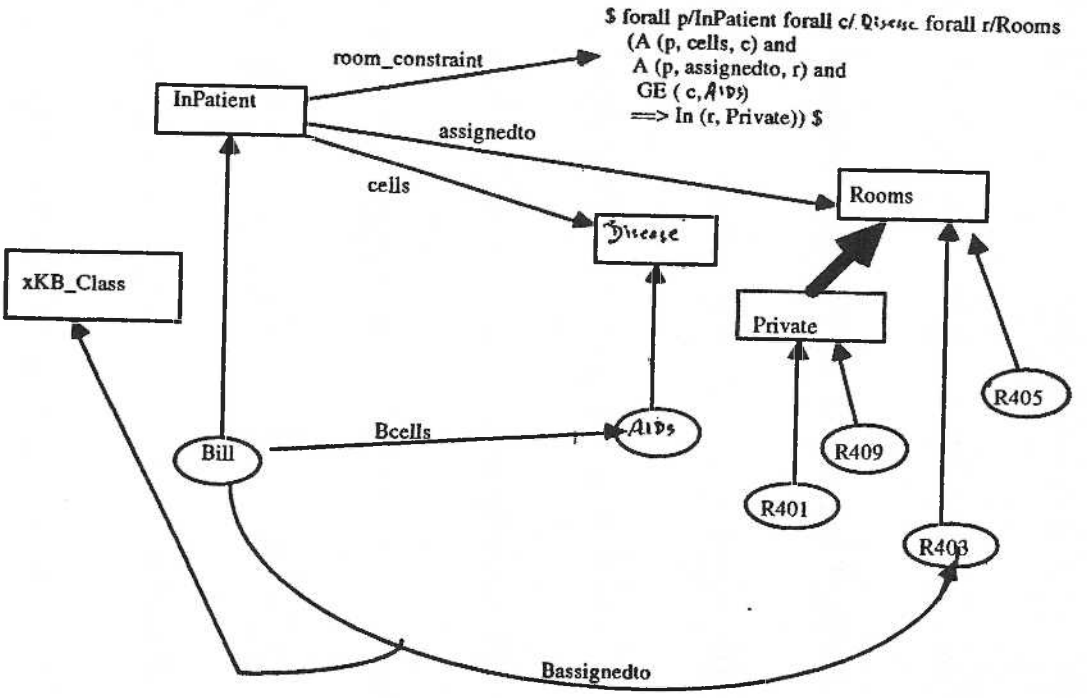
ConceptBase is a information system management system developed at the University of Passau and Aachen, under the direction of Prof. Dr. M. Jarke [Jarke, 1991]. ConceptBase was designed in 1988 with the intention of using it as the "global KBMS" of the DAIDA environment for the information system development of data-intensive applications [Jarke, 1990]. ConceptBase has since been extended and is being used in a wide variety of applications in many locations around the world.

ConceptBase implements the knowledge representation language Telos [Mylopoulos et al. 1990]. ConceptBase operates on SUN workstations under the UNIX operating system. The main implementation language is BIM-Prolog [BIM, 1990] with special interfaces written in C.

The creation of the xKB and its distinction from the full information system (KB) is implemented not in terms of separate physical entities but by means of representing objects in the xKB portion as instances of a specific class, the `xKB_Class` (fig. 6).

According to the xKB approach, whenever the information system is modified the integrity is checked twice, once for this user's subset of the information system and once for the whole information system. The check switch is used to specify whether the integrity check should be carried out only for this user's subset of the information system or for the whole information system. The check switch is used to indicate whether objects retrieved from the information system require access permission in order to be retrieved. If access permission is not required (check switch is off) then all objects in the information system can be accessed and hence the integrity of the whole information system is checked. If access permission is required (check switch is on) then integrity checking takes place only for the subset of the information system which this user may access.

Figure 6. "Invalid" objects represented as instances of the xKB class.



The security task may initiate a process which attempts to move objects in the xKB to the KB portion of the information system. Modifications to the information system at some time after these objects were placed in the xKB may make this possible. The security task may not know which objects exist in the xKB portion of the information system. That is only possible for those tasks who have access to information in the xKB which does not satisfy their integrity constraints.

The modification to the ConceptBase system which gives a user the option to place access permissions to objects in a pending status is implemented by specifying the access permission for such objects with a future validity time. This is possible since ConceptBase has a temporal component, representing validity time as a part of each proposition. The pending objects would not appear in the task list, but be stored as propositions with the future validity time. Users are regularly reminded, currently every time they initiate a session, whether they have objects with a pending status.

If access permission to objects associated with a task are in the pending mode, then no tell operations are possible, i.e., no modifications to the information system can be made. This restriction could be loosened by only prohibiting operations related to the pending objects, although determining related objects appears to be a complex requirement.

Summary

The xKB approach described in this section allows the assignment of security to constraints and provides a method to support multi-user integrity, i.e., integrity in the subset of the information system which each user may access, as well as integrity in the whole information system. Thereby security and integrity can exist together, although with some operating restrictions. Some information is not considered to be a part of the full information system, although users may access it. Users may be limited in their operations if pending information exists.

The xKB approach requires extra time to be spent in integrity checking since the modification of the information system requires integrity checking of both the user's subset of the information system as well as the whole information system. This extra time is dependent on the size of the KB and the size of the xKB. hought should be given to executing the second integrity check, which determines whether the object goes into the KB or the xKB, as a background process, since the user does not know about the process.

To prevent two "divergent" information systems from overwhelming the system, further research must be directed in the area of moving objects from the xKB to the KB. As operations are performed with objects in the xKB, triggers could be released to check whether the object can be moved from the xKB to the KB.

Further research is also required to determine when a user truly needs a information system which satisfies all integrity constraints and the effect of placing objects in a pending state. A well-defined list of operations which are allowed and which are not allowed based on the existence of pending objects is required.

Access permission can be granted to constraints. The accessible constraints must be satisfied in the user's subset of the information system, while constraints that cannot be accessed do not have to be satisfied. The resulting integrity problems are managed by means of the xKB approach which defines an area of the information system for that information which does not satisfy all integrity constraints. Furthermore, users of a task may limit their own accessible information by marking access to information as pending, in order to achieve a meaningful subset of the information system.

The xKB approach allows each user's subset of the information system to be correct, while also maintaining the integrity of the full information system.

References

Berson, T.A. and Lunt, T.F. (1987). Multilevel Security for Knowledge-Based Systems. ***Proc. of the IEEE Symposium on Security and Privacy***, 235-242.

BIM Information Technology (1990). ***ProLog by BIM - 3.0 Reference Manual***, Everberg, Belgium.

Biskup, J. (1990). A general framework for database security. ***European Symposium on Research in Computer Security***, ESORICS, Toulouse, France.

Borigda, A. (1985, Dec.). Language features for flexible handling of exceptions in information systems. ***ACM Transactions on Database Systems***, Vol. 10, No.4, 565-603.

Brodie, M.L. (1988). Future intelligent information systems: AI and database technologies working together. ***Readings in artificial intelligence and databases***, (eds.) M. Brodie, J. Mylopoulos, Morgan Kaufman, San Mateo, CA.

Bry, F. and Manthey, R. (1986). Checking consistency of database constraints: A logical basis., ***Proc. of the 12th International Conference on Very Large Databases***, VLDB, Kyota, 13-20.

F. Bry, Decker,H. & Manthey, R. (1988). A uniform approach to constraint satisfaction and constraint satisfiability in deductive databases. ***Proc. EDBT***, Venice, 488-505.

Chakravarthy, U.S. , Grant J. & Minker J. (1990, June). Logic-based approach to semantic query optimization. ***ACM Transactions on Database Systems***, Vol. 15, No. 2, 162-207.

Denning, D. E. (1988). Lessons learned from modeling a secure multilevel relational database system. ***Database Security: Status and Prospects***, (ed.) C.E. Landwehr, Elsevier, IFIP, 35-43.

Denning, D.E., Lunt, T.F. Schaef, T.F. Lunt, Schell, R., Shchokley, W. and Heckman, M. (1988). The SeaView security model. ***Proc. of the IEEE Symposium on Security and Privacy***, 218-233.

Dittrich, K.R., Hartig, M., and Pfefferle, H. (1989). Discretionary access control in structurally object-oriented database systems. ***Database security II: Status and prospects***, (ed.) C. Landwehr, Elsevier, IFIP, 105-121.

Garvey, T. D. & Lunt, T. F. (1991, Feb.). Multilevel security for knowledge based systems, ***CSL Technical Report***. SRI International, SRI-CSL-91-01.

Garvey, T. D. and Lunt, T. F. (1991, Nov.). Cover stories for database securit. ***Proc.of the Fifth IFIP WG11.3 Workshop on Database Security***, Shepherdstown, W.VA.

Greenberg, I. (1991, April). Distributed database security, ***SRI Project 8772, Final Report***. SRI International.

Haigh, J.T. (1988). Modeling database security requirements. ***Database security: Status and prospects,*** (ed.) C. Landwehr, Elsevier, IFIP, 45-56.

Jarke, M. (1990). DAIDA - Conceptual modeling and knowledge-based support of information systems development processes. ***Technique et Science Informatiques***, Vol. 9, No. 2, 121-133.

Jarke, M. Ed. (1991). ***ConceptBase V3.0 user manual***, MIP-9106, University of Passau.

Keefe, T. F, Thomsen, D. J., Tsai, W. T. & Hansch, M. R. (1989, Dec.). Multi-party update conflict: The problem and its Solutions, ***Fifth Annual IEEE Computer Security Applications Conference***, Tucson, Arizona, 222-231.

Laferierre, C. (1991). Predicate based polyinstantiation in multi-level secure DBMS. ***Computers & Security***, Vol. 10, No. 1, 41-49.

Lunt, T.F. (1990). The true meaning of polyinstantiation. ***Proc. of the Third RADC Database Security Workshop.***

Lunt ,T.F. (1991, June). Polyinstantiation: An inevitable part of a multilevel world. ***Proc. of the IEEE Computer Security Foundations Workshop IV***, Franconia, NH, 236-238.

Lunt, T.F. and Hsieh, D. (1991). Update semantics for a multilevel relational database system. ***Database security IV: Status and Prospects***, (eds.) S. Jajodia, C. Landwehr, IFIP, 281-296.

Lunt, T., Schell, R. R., Shockley, W. R., Heckman, M., Waren, D. (1988). A near-term design for the SeaView multilevel database system. ***Proc. of the Symposium on Security and Privacy***, 234-244.

Meadows, C. & Jajodia, S. (1988). Integrity versus security in multi-level secure databases, ***Database security: Status and prospects***, (ed.) C.E. Landwehr, Elsevier, IFIP, 89-101.

Morgenstern, M. (1987). Security and inference in multilevel database and knowledge-base systems. ***Proc. ACM SIGMOD***, 357-373.

Mylopoulosk J., Borgida, A., Jarke, M., Koubarakis, M. (1990, Oct). Telos: Representing knowledge about information systems. ***ACM Transactions on Information Systems***, Vol. 8, No. 4, 325-362.

Schell, R. R. and Denning, D. E. (1986). Integrity in trusted database systems. ***Proc. of the 9th National Computer Security Conference***, 30-36.

Steinke, **G.** (1991). Task-based security for knowledge base systems. ***Reihe Informatik***, Verlag Shaker.

Summers, R. C. & Kurzban, S. A. (1988). Potential applications of knowledge-based methods to computer security. ***Computers & Security***, Vol. 7, No. 4, 373-385.

Wiseman, **S.** (1990). Control of confidentiality in databases. ***Computers & Security***, Vol. 9, No. 6, 529-537.