Journal of Computer Science 2 (1): 19-28, 2006
ISSN 1549-3636
© 2006 Science Publications

Towards a Temporal Multilevel Secure Database (TMSDB)

Ramzi A. Haraty and Natalie Bekaii
Division of Computer Science and Mathematics
Lebanese American University, Beirut, Lebanon

Abstract: Standard Relational Databases are used to store the state of reality at a single moment of time. Temporal Databases are used to store Time-Varying Data. Multilevel Secure Databases are used to securely store highly sensitive data. Each of these databases serves its purpose well, but if we were to model the temporal and sensitive aspect of the real world data, we will not be able to use any of the previously mentioned databases. Our aim here is to develop a new database model that can be used to model multilevel secure temporal data.

Key words: Temporal multilevel secure database, standard relational database, temporal database

INTRODUCTION

Conventional relational databases were designed to capture reality. They present some aspect of the real world. They were designed to keep record of current data without keeping track of historical data. In conventional databases, changes to the real world are reflected in the database. Old data stored in the database is deleted or updated every time a new change occurs. On the other hand, temporal databases which are another type of databases are by definition databases that keep track of historical data. Old data is never deleted or updated. These databases can be used to capture past, present and future data, but they provide little support for insuring the secrecy of data. Multilevel secure database a third type of databases, insure the secrecy of data and insure that each user only gains access to only those data for which he/she has proper clearance, but do not support the recording and querying of time varying data, or historical data. Our aim in this study is to present the definition of a new relational database model, a model that combines both the properties of the temporal database model and the multilevel secure database model, a model that supports both recording of secret and temporal data. The new-presented model is the temporal multilevel secure database model.

Historical background: Several relational models for multilevel secure databases have been proposed over the years. All of these proposals implement the policy of mandatory access protection defined[1]. Mandatory access controls policies were interpreted for computerized systems. The mandatory access policies were made to control access to highly sensitive data. They are used in applications that support data with different access classes and users with different authorizations, applications such as civilian, military, commercial and governmental agencies. In these applications the objects (data items) are grouped by their classification and the subjects (active processes, users) are grouped by their clearance level.

Accesses to objects are allowed or denied based on a comparison of the classification associated with the object and clearance associated with the subject. We take the classifications and clearances from a domain of partially ordered access classes. For two access classes c1 and c2, c1 is higher than c2 if $c1 > c2$ and c1 dominates c2 if $c1 \geq c2$. Mandatory access policies state that no direct access by unauthorized user to classified data is to be allowed. They also enforce secure information flow by preventing information from flowing indirectly from high access classes to lower access classes.

According to Bell-LaPadula two restrictions are imposed on all data accesses.

* The simple security property: A subject is allowed a read access to an object only if the subject clearance is identical to or higher than the object's classification.
* The ★-property: A subject is allowed a write access to an object only if the subject's clearance is identical to or lower than the object's classification.

To implement the two restrictions listed above, access classes were associated with the elements of a relation and clearances were associated with subjects accessing the relation. Subjects with different clearances see different versions of a multilevel relation.

In addition to normal security threats, there were some indirect means that threatened the security of the system, a problem referred to as the covert channel. Covert channels result from passing down information

Corresponding Author: Ramzi A. Haraty, Lebanese American University, P.O. Box 13-5053 Chouran, Beirut, Lebanon 1102, 2801

indirectly by subjects at high levels to subjects at lower levels. Let us look at the following scenario: A subject at a level C wants to insert a new tuple with a primary key pk at level C and it happens that a tuple with the same primary key pk is inserted at the higher level S by a subject S. Taking this case from a database perspective, the insertion of the new tuple should be rejected, but with respect to security, to reject the insertion of this tuple will infer indirectly to the user at the level C the existence of a tuple with the same primary key pk at a higher security level. To avoid this problem, multilevel relations had to be allowed to contain multiple tuples with the same primary key and these tuples were known as the polyinstantiated tuples. There are two different types of polyintantiation was defined[2]:

* Entity or Tuple Polyinstantiation: is to allow multiple tuples with the same primary key but different access classes to be stored within the same relation.
* Attribute or Element Polyinstantiation: is to allow two or more tuples with the same primary key and same associated access classes but with different values for one or more of the remaining non-key attributes to be stored in the same relation.

The coexistence of two or more tuples with the same primary key in the same relation will result in ambiguity and confusion for users. Therefore, some additional integrity constraints needed to be specified to be able to control polyinstantiation and to avoid data ambiguity.

In general all the multilevel relational data models were based on the mandatory access policies and the concept of tuple polyinstantiation, as for the additional integrity constraints they differed from one data model to another, some were alike, some were based on those defined in other models and some were totally different. The Sea View model[3] was the first model to implement the mandatory protection policies. This model was developed by SRI International and Gemini Computers. The Sea View model implement multilevel relations using an algorithm called the decomposition algorithm. The decomposition algorithm decomposes multilevel real relations into single-level base relations. Later a recovery algorithm is used to recover a multilevel real relation from a single-level relation.

The Jajodia-Sandhu model was derived from the Sea View model. The Jajodia-Sandhu model discussed the most fundamental aspects of the multilevel relational data model independently of implementation issues. Many aspects of the Jajodia-Sandhu model are derived from the Sea View model. What Jajodia-Sandhu model added to the Sea View model is the requirement that at each access class there can be at most one tuple for each entity. In Jajodia-Sandhu model modified versions of the Sea View decompositions and recovery algorithms were given.

The LDV model[4] is another multilevel secure relational database model. In the LDV model some restrictions are placed on polyinstantiation. To allow tuple polyinstantiation in multilevel relations, a maintenance level is associated with each tuple in the database. The maintenance level of a tuple is the level at which the tuple was inserted into the database. The strength of the LDV model is based on the derivation technique used to solve element polyinstantiation and the classification constraints, used to solve covert channels.

The MLR data model[5] is a model that combines ideas from Sea View, belief-based semantics and LDV model. It is a simple, unambiguous and has the advantage of retaining upward information flow. Moreover it has five integrity constraints and five operation statements for manipulating multilevel relations.

The MLR data model retained some previously defined concepts such as polyinstanstiation, referential integrity and data manipulation concepts and introduced several new concepts such as the data-borrow integrity and the uplevel statement.

Standard relational data models were designed to store a snapshot of the real world at a single instant of time. In these databases the variation of data over time is treated the same way as ordinary data. This might be just what we need in certain applications, but in other applications where there is a need to store the past, present and future of data this is not enough. These applications include scheduling applications such as airlines, trains, record-keeping applications such as medical applications, accounting, banking and inventory applications. The use of standard relational databases for these applications will cause a high data redundancy problem. The demand for storing and managing historical and time varying data started to appear in the early 1970s in the area of medical information systems; this interest increased in 1982. A bibliography contained 80 articles from the years 1982 till 1986, was published in 1986[6]. In 1986, there were at least 25 groups studying time in databases. Among these groups we have Ben-Zvi[7], Ariav *et al.* [8], Snodgrass[9], Lum *et al.*[10], Clifford[11], Snodgrass and Ahn[12] and Gadia and Vaishmav[13]. These studies can be classified into three categories: the formulation of semantics of time at the conceptual level, the development of a model for temporal databases and the design of temporal query languages.

The studies done to develop a temporal data model followed two approaches.

The first approach is to extend the standard relational data model so that it supports time varying data and the second approach is based on extending the snapshot model with time appearing as additional attributes.

A book published in year 1993[14], was one of the books that extensively covered temporal databases research till year 1993. It is an excellent reference to the different temporal databases models published till that year.

The Historical Relational Data Model (HRDM) was one of the earliest temporal database models. This historical model is a consistent extension of the traditional relational data model. It supported the recording of time varying data, modeled relationships over time and enforced referential integrity constraints with respect to the added temporal dimension. This model added the new object, the set T of times, to the standard relational data model and changed the domains of the relation attributes to become functions from points in time (T) into some simple value domain.

Users over the years begun to increasingly request for temporal database models, therefore, the time dimension has been added to many data models. These models include the entity-relationship model, semantic data models, knowledge based data models, relational data model, object-oriented data model and deductive databases.

In the past few years, more than 2000 papers and books have been written about temporal databases. Most of these papers were listed in a series of seven cumulative bibliographies (the newest one[15] provides pointers to its predecessors). Other bibliographies with papers talking about temporal databases published in the last seven years.

The first book that entirely talked about temporal databases was published in year 2000 "Developping Time-Oriented Database Aplications in SQL" [16].

Reviewing the historical background of temporal databases and multilevel secure databases[17] is a paper that investigates the applicability of the parametric model for temporal data to query multilevel security data. This paper gives a brief introduction to the parametric model for temporal data and the WSQ model for multilevel security database and shows how to adapt the parametric model to multilevel security. The Concept of a user hierarchy in the parametric model is introduced. As an example for the user hierarchy, let us consider the following community of users: system, public, analyzer and classical with user domains [0, NOW], [0,NOW - 10], [NOW - 4,NOW] and {NOW}, respectively. The system user can see the whole information, the public can only see information at least 10 years old, the analyzer has the last 5 years worth of information and the classical user only sees the current information. This study introduces the two models for multilevel security the parametric model and the WSQ model and shows that queries can be expressed more naturally in the parametric model.

MULTILEVEL SECURE DATABASES

Multilevel secure databases are databases that contain large amounts of very highly sensitive and confidential data (e.g., military, governmental, etc.…) that is why access to the data stored in these databases needs to be authorized. Although, there is no clear agreement on the definition of a multilevel secure database model, we try to present in this chapter the basic concepts of a multilevel secure relational model. Our aim is to use the fundamental aspects presented in this chapter, in building up the model for temporal multilevel secure databases.

One of the main concepts in multilevel secure databases is the assignment of access privileges to users of the database so as to be able to manage and protect confidential and sensitive data. Each user is given access privileges to access the data he/she is authorized to access. We protect confidential data either by making it inaccessible to unauthorized users or by providing a cover story. To provide a cover story, the same real-world entity is depicted by more than one record. Each of these records is assigned a different level classification. Users with different access clearances see different versions of the data in the database. These records have the same primary key at all the classification levels but with different values for the non-key attributes at each classification level. This technique is used to protect information stored at a higher security level by providing some lower security levels. Data hidden from lower clearance users will be seen by a user of a higher clearance if this user has the clearance to see this data.

Access privileges can be assigned to relations, to individual tuples in a relation, to individual columns, or to individual data elements of a relation.

TEMPORAL DATABASES

A standard relation is two-dimensional with attributes and tuples as dimensions. A temporal relation contains two additional, orthogonal time dimensions, namely valid time and transaction time. Valid time denotes the interval of time during which according to our beliefs a fact is true with respect to the real world. Transaction time records when facts are stored in the temporal relation. Valid and transaction time have precise, crisp definitions. If changes to the past are important, then valid time support is required. If it is necessary to rollback to a previous state of the database, then transaction time support is called for. Moreover, valid times can be updated since they reflect our beliefs of when a tuple is considered to be true, but transaction times can not be updated, since they reflect the time a tuple is recorded in the database and this time is set by the system and not by the user and therefore it can be changed.

Temporal Databases can be divided into three types based on the two different types of time dimensions used in temporal databases.

* Transaction Time Databases: Transaction Time database is a database that contains only one of the

two orthogonal time dimensions, the transaction time. These databases support the recording of past and present data only.

* Valid Time Databases: Valid Time database is a temporal database that records only the valid time orthogonal time dimension. It supports the recording of past, present and future data, since the valid time depends on what the user believes.
* Bitemporal Databases: The Bitemporal database is a database that contains both the two time orthogonal dimensions, the valid time and the transaction time. It supports the recording of past, present and future data. Not only it supports the recording of what we believe was true, is true, or will be true but also the recording of the time of when we did believe so.

In temporal databases other than the primary and foreign key constraints we have two main constraints. The first constraint is used to solve the redundancy and circumlocution problems and the second is used to solve the contradiction problem.

* Redundancy and circumlocution problems Constraint: If at any time a relation contains two distinct tuples that are identical except for their valid time values i1 and i2 ➔ i1 merges i2 must be false.
* Contradiction problem Constraint: If at any given time a relation containing two tuples that have the apparent primary key value but differ on the values of their non-key attributes then their valid time values i1 and i2 must be such the i1 overlaps i2 is false.

TEMPORAL MULTILEVEL SECURE DATABASES MODEL

This study brings together two research directions in databases technology, temporal databases that keeps record of the history of data and multilevel secure databases which groups data in a database into different classification levels and allows only users with the appropriate security clearance to access the data stored on the corresponding classification level. Over the last twenty years, there ha been a major demand for recording historical data and in the past few years, the concern for data security has increased. Temporal multilevel secure databases meet the two requirements. They are concerned with assigning access privileges to past, current and future data. They have both the characteristics of a temporal database and those of a multilevel secure database. In this chapter, we present the definition for a temporal multilevel secure database model. The definition of a temporal multilevel secure relation is presented below:

A temporal multilevel secure relation is of the form:

R(A1,C1,A2,C2,…,An,Cn,VT,Cvt,TC)
Where, Ai is a data attribute over domain Di, Ci is a classification attribute for Ai, VT is the valid time attribute, Cvt is a classification attribute for the valid time attribute and TC is the tuple-class attribute. The domain of Ci is specified by a set {Li, …,Hi} which enumerates the allowed values for access classes, ranging from the greatest lower bound (glb) Li to the least upper bound (lub) Hi. The domain of TC is the set {lub{Li; i=1,…,n},…, lub{Hi: i= 1,…,n}} and the domain of Cvt is the set {lub{Li; i=1,…,n},…, lub{Hi: i= 1,…,n}.

In multilevel temporal databases, we store different database states and users with different clearances see different versions of these database states. These different versions must be kept coherent and consistent, without introducing any downward signaling channels. All the tuples in the database must be meaningful, so we should not have redundancy, circumlocution or contradiction problems. To be able to meet all of these requirements we need to specify some constraints on temporal multilevel secure databases. These constraints must be a combination of the integrity constraints of temporal databases along with those of multilevel secure databases.

Entity integrity: Let AK be the apparent key of R and let VT be the valid time of R, A temporal multilevel relation R satisfies entity integrity if and only if for all instances Rc of R and $t \in Rc$:
$Ai \in AK \Rightarrow [Ai] \neq null$
$[VTi] \neq null$
$Ai, Aj \in AK \Rightarrow t[Ci] = t[Cj] = t[CVT]$ (where CVT is the classification of the valid time)
$Ai \notin AK$ and $Ai <> VTi \Rightarrow t[Ci] \geq t[CAK]$ (where CAK is the classification of the apparent key)

The first requirement ensures that no attribute of a primary key of a base relation may be null. The second requirement specifies that the valid time value can never be null. The third requirement ensures that all the attributes of a primary key of a base relation must have the same access class, not only this the valid time access class must also have the same access class as these attributes. The fourth requirement says that the access class of all non-key attributes (the valid time is not included) in a tuple dominates the access class of the primary key.

Null integrity: A multilevel temporal relation R satisfies null integrity if and only if for each instance Rc of R both of the following conditions are true:
For all $t \in Rc$, $t[Ai] = null \Rightarrow t[Ci] = t[CAK]$;

Let us say that tuple t subsumes tuple s if for every attribute Ai, either

t[VTi] overlaps s[VTi] and t[Ai] ≠ s[Ai]

or

t[Ai] = s[Ai] and t[VTi] merges s[VTi]

The first requirement means that attributes that have null values have an access class that is equal to the access class of the primary key. The second requirement states that Rc does not contain two distinct tuples with different non-key attributes values and the valid time of one overlaps the valid time of another, or two distinct tuples with identical value for all the attributes and the valid time of one merges the valid time of another. Having such tuples will lead to a problem similar to one of the problems we had in temporal relational model, the redundancy, circumlocution or contradiction problem. That is why we need to prevent the existence of such tuples either by combining the tuples that have a redundancy or circumlocution problem or by preventing the existence of tuples that would cause contradiction (Note we are talking about the attributes that would an access class similar to that of the primary key and therefore similar to that of the valid time).

InterInstance integrity: R satisfies interinstance integrity if and only if for all c' ≤ c we have Rc' = σ (Rc, c'), where the filter function σ produces the c' instance Rc', from Rc as follows:

For every tuple t ∈ Rc such that t[CAK] = c' , there is a tuple t' ∈ Rc' with t'[AK,CAK] and for Ai ∉ AK

$$t'[Ai,Ci] = \begin{cases} t[Ai,Ci] \text{ if } t[Ci] \le c' \\ \text{<null,t[CAK]> otherwise} \end{cases}$$

There are no tuples in Rc' other than those derived by the above rule.

If at any given time the end result contained two tuples that have the same apparent primary key value (the non valid time attributes of the primary key) but differ on the values of their non-key attributes then their valid time values i1 and i2 must be such that i1 overlaps i2 is false.

If at any time the end result contained two distinct tuples that are identical except for their valid time values i1 and i2, then i1 merges i2 must be false.

In this constraint, the filter function is used to map the multilevel temporal relation to different instances, one for each access class, so as to give the user the ability to see only the historical data for which he is cleared. The resulting obtained instance will be similar in a way to a temporal database. In addition, we must make sure in the end result to combine the tuples that cause redundancy or circumlocution and not to have two tuples that lead to a contradiction.

Polyinstantiation integrity: In temporal multilevel secure databases, we may have several tuples with the same primary key but with different values for the non-key attributes. Not only this, even at the same access level we will have more than one tuple with the same primary key but with different valid times. As previously mentioned in multilevel secure databases we cannot prevent a low user from inserting a tuple with the same primary key as a previously inserted high level tuple or we might create some downward signaling channel that will violate the secrecy of data. At the same time we can not prevent a user at the same access level from inserting a tuple with the same primary key as an old existing tuple at the same access level but with different valid time. We can either refuse such an insertion or override existing data. Refusing to insert this tuple, or overriding existing data for either any of the two previously mentioned reasons will cause a downward signaling channel, the loss of secret information and the destruction of historical data. We have no choice but to keep all the tuples without violating the foundations of relational databases. That's why we need to declare the access class and the valid time to be part of the primary key. So we need to specify the following key constraint:

R satisfies the key integrity if and only if for every Rc we have for all Ai: AK, VT, CAK, CVT, Ci → Ai. that means that the user specified primary key AK in conjunction with the valid time, the classification attributes CAK, the classification attribute CVT and Ci, functionally determines the values of Ai attribute.

The operations on a relational database can be categorized into two main categories retrievals and updates. The update operations can be divided into three types of operations: Insert which is used to insert a new tuple or tuples in a relation, Delete which is used to delete tuples and Modify which is used to change the values of some attributes.

We only need to worry about insert and modify operations in a temporal multilevel secure database. We are going to show by examples how update operations take place in a temporal multilevel secure database. Whenever we need to do an update operation, we need to make sure not to violate the integrity constraints specified on the database.

Let us take for example the temporal multilevel military officers relation presented in Table 1.

Table 1: An example of a temporal multilevel secure relation

ID	Name	Rank	Valid Time	TC
100 U	Johnson U	Major General U	[1953/3-∞] U	U
101 S	Miles S	Marshal S	[1985/7-∞] S	S
ID	Name	Rank	Valid Time	TC
100 U	Johnson U	Major General U	[1953/3-∞] U	U
101 S	Miles S	Marshal S	[1985/7-∞] S	S

When introducing multilevel access classes. Access clearances are assigned to individual data elements of a relation. Subjects having different clearances see different versions of the military officers' relation. A U-User having a clearance at the access class u will see a version of the military officers' relation that includes only the data that were assigned an access class u. While an S-User will be able to see a version of the military officers table that will include both the data that were assigned an access class u and an access class s. In order to be able to record time varying data into our database we need to extend the military officers' relation by adding the temporal attribute ValidTime. It is an interval that we use to determine when the data inserted into the tuple, was, is or will be valid. The U-user version of the military officers table is shown in Table 2 and the S-User version of the military officers table is shown in Table 3.

Table 2: The U-user version of the military_officer's table

ID	Name	Rank	Valid Time	TC
100 U	Johnson U	Major General U	[1953/3-∞] U	U

Table 3: The S-user version of the military _officer's table

ID	Name	Rank	Valid Time	TC
100 U	Johnson U	Major General U	[1953/3-∞] U	U
101 S	Miles S	Inspector General S	[1985/7-∞] S	S

Table 4: The U-user version of the military_officer's table

ID	Name	Rank	Valid Time	TC
100 U	Johnson U	Major General U	[1953/3-1981/3] U	U
100 U	Johnson U	Lieutanent General U	[1981/4-∞] U	U

In order to understand how update operations take place in a temporal multilevel secure relation, let us take the following example. Let us assume that on April 1981 a U-User wants to update the rank of the military officer "Johnson" from "Major General" to "Lieutenant General". Table 4 shows the U-user version of the military officers table and Table 5 shows the S-User version after this update.

Table 5: The S-user version of the military_officer's table

ID	Name	Rank	Valid Time	TC
100 U	Johnson U	Major General U	[1953/3-1981/3] U	U
100 U	Johnson U	Lieutanent General U	[1981/4-∞] U	U
101 S	Miles S	Inspector General S	[1985/7-∞] S	S

As a result to this update a whole new tuple is inserted. This new tuple is inserted at the U class. The valid time for the old tuple of the military officer "Johnson" is updated to reflect the time in history when the rank of officer "Johnson" was "Major General". From the date April 1981, the rank of the officer "Johnson" changed to "Lieutenant General".

As we can see from that example, an update performed by a User with an X clearance on a tuple with an access privilege X is dealt with in a way similar to the way we deal with an update operation in a temporal database by inserting a new tuple with a new valid time interval and updating the valid time interval of the old tuple. The new inserted tuple will also have an access privilege X.

Let us take another example, in which we deal with the case where a higher level user tries to update a tuple with a lower level access privilege. Going back to our Military Officers example, a tuple originally inserted by a U-User to the Military Officers table, can be updated by a higher level user like the S-User. Assume that on February 1, 1997, a user with an S clearance gives the two officers "Johnson" and "Miles" a new higher rank. Since in temporal databases whenever we are updating we do not actually update the value, but we rather insert a new tuple with the same values for all the attributes, except for the attribute that is being updated and the valid time timestamp value, the same applies here and the S user would have to insert a new tuple. This tuple would be inserted at the S-level since an S-user is performing the operation (Table 6 and 7). But this would create a problem, because we would have two tuples with the same apparent key with overlapping timestamps.

Table 6: The S-user version of the military_officer's table

ID	Name	Rank	Valid Time	TC
100 U	Johnson U	Major General U	[1953/3-1981/3] U	U
100 U	Johnson U	Lieutanent GeneralU	[1981/4-∞] U	U
100 S	Johnson S	Inspector General S	[1997/2-∞] S	S
101 S	Miles S	Inspector General S	[1981/4-1987/1] S	S
101 S	Miles S	Marshal S	[1997/2-∞] S	S

Table 7: The U-user version of the military_officer's table

ID	Name	Rank	Valid Time	TC
100 U	Johnson U	Major General U	[1953/3-1981/3] U	U
100 U	Johnson U	Lieutanent General U	[1981/4-1997/1] U	U
100 U	Johnson U	Null U	[1997/2-∞] U	U

Table 8: The S-user version of the military_officer's table

ID	Name	Rank	Valid Time	TC
100 U	Johnson U	Major General U	[1953/3-1981/3] U	U
100 U	Johnson U	Lieutanent General U	[1981/4-1997/1] U	U
100 U	Johnson U	Null U	[1997/2-∞] U	U
100 S	Johnson S	Inspector General S	[1997/2-∞] S	S
101 S	Miles S	Inspector General S	[1981/4-1987/1] S	S
101 S	Miles S	Marshal S	[1997/2-∞] S	S

This would result in a temporary inconsistency in the database that needs to be resolved. For instance, the inconsistency may be resolved as follows: The S-user logs on at the U level and insert a new tuple with a nullified rank value that happens to have the same timestamp of the tuple inserted at the S-level. Table 7 and 8 show the results.

This schema won't create a downward signaling channel from one subject to another. Since the nullification of the salary at the U-level is being done by a U-subject. Someone might say that there is a downward signaling channel with a human in the loop. The human is, however trusted not to let the channel be exercised without good cause.

The coexistence of the tuple (100, Johnson, Inspector General, [1997/2-∞]) and the tuple (100, Johnson, null,[1997/2-∞]) in Military OfficersS , two tuples with the same primary key, is what we call polyinstantiation. Here there is no threat of entity or attribute polyinstantiation, because at any time the attribute value is updated this means that a new tuple would need to be inserted with the same primary key, same time timestamps, but with different value for the attribute at each level, the value of the attribute would appear null at the lowest level, if this attribute was updated by a higher level user.

Another problem that the coexistence of these two tuples might create is that they both have the same time timestamps. In temporal databases at any given instance of time each military officer is supposed to have only one rank. This is a problem that we refer to as the contradiction problem. Since the military officer 100 is shown to have a rank of both null and "Inspector General" from the date February1997 and up to this date.

There is another option to consider when dealing with the problem of having two tuples with the same primary key and overlapping times.The S user could login at the U level but instead of inserting a new tuple with a nullified rank value that happens to have the same timestamp as the tuple inserted at the S-level (Table 7 and 8) he could just perform a temporal delete on the military officer '100'. This delete won't result in deleting the military officer '100' at the U-level, but it would rather end the validity time of this tuple by changing the end value of the valid interval to the start value of the valid interval of the tuple inserted at the S level, decreased of one unit of the used temporal granularity. Table 9 shows the U_user version of the military officers table with the ended valid time for officer '100' and Table 10 shows the S-user version of the military officers table.

Table 9: The U-user version of the military_officer's table

ID		Name		Rank		Valid Time		TC
100	U	Johnson	U	Major General	U	[1953/3-1981/3]	U	U
100	U	Johnson	U	Lieutanent General	U	[1981/4-1997/1]	U	U

The deletion of the officer '100' U-user tuple will not result in a downward signaling channel since the tuple deletion is done by a U-user and the deletion itself might be considered as a good cover story to the change of the rank of the officer.

MODEL MODIFICATIONS

After taking several examples of the update operations that might take place in a temporal multilevel secure database, we notice that the classifications of all the attributes within a certain tuple are the same. The case where we might have two different classifications for two different attributes in the same tuple will not occur in temporal multilevel secure databases. In multilevel secure databases this case occurs, when a user at a certain level for example level1 inserts a new tuple, then a user at level 2 where level 2 > level 1 updates one of the attributes of the tuple inserted at level 1. In temporal secure databases, the previous example will result in a new tuple inserted at level 2, therefore all of the attributes of the tuple at level 1 will have the same original element classification. Therefore, there is no need of recording the element classification for each attribute, the tuple classification will be sufficient to record the classification of all the attributes. Table 11 shows the S-User version of the military officers table after removing the element classification for each attribute and relying on the tuple classification to record the classification of all the tuple attributes.

Table 11: The S-user version of the military_officer's table

ID	Name	Rank	Valid Time	TC
100	Johnson	Major General	[1953/3-1981/3]	U
100	Johnson	Lieutanent General	[1981/4-1997/1]	U
100	Johnson	Inspector General	[1997/2-∞]	S
101	Miles	Inspector General	[1981/4-1987/1]	S
101	Miles	Marshal	[1997/2-∞]	S

The removal of the element classification will lead to a change in the definition of the temporal multilevel secure database. The new definition of the temporal multilevel secure relation would be as follows:

R (A1,A2,…,An, VT, TC)

Where, Ai is a data attribute over domain Di, VT is the valid time attribute and TC is the tuple-class attribute. The domain of TC is specified by a set {Li, …,Hi} which enumerates the allowed values for access classes, ranging from the greatest lower bound (glb) Li to the least upper bound (lub) Hi.

Since the definition of the temporal multilevel secure relation has been changed so would the definitions of the integrity constraints. The new definitions are listed below:

Entity integrity: Let AK be the apparent key of R and let VT be the valid time of R, A temporal multilevel relation R satisfies entity integrity if and only if for all instances Rc of R and $t \in$ Rc:
$A_i \in AK \Rightarrow [A_i] \neq null$
$[VT_i] \neq null$

The new definition of the entity integrity constraint specifies that no part of the primary key can have a null. The first requirement ensures that no attribute of a primary key of a base relation may be null. The second requirement specifies that the valid time value can never be null. The Third and the fourth requirement have been removed of the new definition.

Null integrity: A multilevel temporal relation R satisfies null integrity if and only if for each instance Rc of R both of the following conditions are true:

Let us say that tuple t subsumes tuple s if for every attribute Ai, either
t[VTi] overlaps s[VTi] and t[Ai] ≠ s[Ai]
or
t[Ai] = s[Ai] and t[VTi] merges s[VTi]

The first requirement of this constraint is no longer needed. The second requirement that states that Rc does not contain two distinct tuples with different non-key attributes values and the valid time of one overlaps the valid time of another, or two distinct tuples with identical value for all the attributes and the valid time of one merges the valid time of another, still holds.

InterInstance integrity: R satisfies interinstance integrity if and only if for all c' ≤ c we have Rc' = σ (Rc, c'), where the filter function σ produces the c' instance Rc', from Rc as follows:
For every tuple t ∈ Rc such that TC = c', there is a tuple t' ∈ Rc' with t'[TC]
There are no tuples in Rc' other than those derived by the above rule.

If at any given time the end result contained two tuples that have the same apparent primary key value (the non valid time attributes of the primary key) but differ on the values of their non-key attributes then their valid time values i1 and i2 must be such that i1 overlaps i2 is false.

If at any time the end result contained two distinct tuples that are identical except for their valid time values i1 and i2, then i1 merges i2 must be false.

The first requirement of this integrity has been modified to meet the new definition of the temporal multilevel secure database relation.

Polyinstantiation integrity: The new definition of the polyinstantiation integrity to meet the new definition of the temporal multilevel secure database. We declare the apparent key, the tuple access class and the valid time to be the primary key. So the new key constraint definition is as follows:

R satisfies the key integrity if and only if for every Rc we have for all Ai, VT, TC → Ai. that means that the user specified primary key AK in conjunction with the valid time, the tuple classification attribute TC functionally determines the values of Ai attribute.

MODEL IMPLEMENTATION

A prototype implementing the concepts presented in this study has been developed. TMSDB, the new developed prototype was used to create and manipulate temporal multilevel secure databases. It was implemented using Powerbuilder 9.0 for application development and Sybase Adaptive Server Anywhere as a DBMS. TMSDB is a GUI application that works as a front end to a relational DBMS. It translates temporal multilevel secure SQL statements into standard SQL statements.

TMSDB is a software application that provides the user with the functionalities of creating and manipulating a temporal multilevel secure database. Using TMSDB the user can create a database, its tables and define primary key and foreign key constraints on these tables. At the physical level, newly created records are time stamped with valid time periods by adding two additional attributes to the table "valid time start" and "valid time end". As for multilevel security a third additional attribute is added to the table, the TC attribute. This attribute is used to record the tuple classification.

Moreover, TMSDB supports temporal query language ATSQL2. ATSQL2 is an extension to SQL and it supports temporal SQL statements and queries, temporal inserts, update and delete statements. The user can query and update the created database using ATQL2. As for queries, the retrieved data will be filtered according to the time flags specified in the temporal SQL statements.

TMSDB also supports multilevel security by the use of a security system that requires the user to login to the system using a login in name and an encrypted password. The login name and password are used to determine the access level of the user. The user access level is used to determine the tuple classification of newly inserted tuples and the filtering criteria to use when returning the results of executed queries.

We provide a list of example queries executed using TMSDB. These queries and their results are listed below.

Example 1: To create the "officer" table the SQL presented below was executed in the TMSDB application.

```
CREATE TABLE officer
 (officer_code varchar(3) not null, officer_f_name varchar(20), officer_l_name varchar(20), primary key(officer_code)) AS VALIDTIME
```

Example 2: The SQL presented below was used to insert a new officer record in the "officer" table. This SQL was executed by an S-user. The result of this SQL is the tuple presented in Table 12.

```
VALIDTIME PERIOD [1990/01/01-2001/01/01)
INSERT INTO "officer"
("officer_code","officer_f_name", "officer_l_name")
VALUES ( '40', 'Thomas', 'Johnson');
```

Table 12: The results of example 2

code	f_name	l_name	Valid Time	TC
40	Thomas	Johnson	[1990/01/01-2001/01/01]	S

Example 3: The SQL presented below was used to insert a new officer record in the "officer" table. This SQL was executed by a U-user. So now we have two

Officers with the same officer code in the Officer table. The two officers' records were inserted by users with different access levels. The table records are presented in Table 13.

```
VALIDTIME PERIOD [1992/01/01-2006/01/01)
INSERT INTO "officer"
("officer_code","officer_f_name", "officer_l_name")
VALUES ('40', 'John', 'Frank');
```

Table 13: The results of example 3

code	Of_name	l_name	Valid Time	TC
40	Thomas	Johnson	[1990/01/01-2001/01/01]	S
40	John	Frank	[1992/01/01 -2006/01/01]	U

Example 4: The SQL presented below was used to insert a new officer record in the "officer" table. This SQL was executed by an S-user. The table records of Officer table are presented in Table 14.

```
VALIDTIME PERIOD [2002/01/01-Forever)
INSERT INTO "officer"
("officer_code","officer_f_name", "officer_l_name")
VALUES ('50', 'Fred', 'Wagner');
```

Table 14: The results of example 4

code	Of_name	l_name	Valid Time	TC
50	Fred	Wagner	[2002/01/01 – Forever]	S
40	Thomas	Johnson	[1990/01/01-2001/01/01]	S
40	John	Frank	[1992/01/01 -2006/01/01]	U

Example 6: A U-user logs on to the TMSDB applications and executes the following SQL query:

```
VALIDTIME PERIOD [beginning – forever)
SELECT  *
FROM officer
```

The Results of the executed SQL query are presented in Table 15. The Only record that will be visible for the U-User is the officer John Record.

Table 15: The results of example 6

code	Of_name	l_name	Valid Time	TC
40	John	Frank	[1992/01/01 -2006/01/01]	U

Example 7: An S-user logs on to the TMSDB applications and executes the same previous SQL query:

The Results of the executed SQL query are presented in Table 16. The Only record that will be visible for the U-User is the officer John Record.

Table 16: The results of example 7

code	Of_name	l_name	Valid Time	TC
50	Fred	Wagner	[2002/01/01 – Forever]	S
40	Thomas	Johnson	[1990/01/01-2001/01/01]	S
40	John	Frank	[1992/01/01 -2006/01/01]	U

Example 8: The S-User wants to modify the Officer "Fred Wagner" family name to "Fred Steinberg". Today's date is '2005/05/04', the result of this update is presented in Table 17. A new tuple is inserted for the officer 'Fred' with the same apparent primary key and same first name, the last name has been changed to 'steinberg', the valid time start is set to today's date and the validtime end value is set forever (to say that the data in this record is valid until changed). The old record for the officer 'Fred' is not modified except for the value of the valid end time which is set to yesterday's date.

```
VALIDTIME PERIOD [2003/01/01 – forever)
UPDATE officer SET  officer_l_name = 'Steinberg'
where officer_code = '50'
```

Table 17: The results of example 8

code	f_name	l_name	Valid Time	TC
50	Fred	Wagner	[2002/01/01 – 2005/05/03]	S
40	Thomas	Johnson	[1990/01/01-2001/01/01]	S
40	John	Frank	[1992/01/01 -2006/01/01]	U
50	Fred	Steinberg	[2005/05/04 – Forever]	S

CONCLUSION

In this study we introduced a new database model the temporal mulltilevel secure relational data model. It is a model that combines the characteristics of the temporal database model and the multilevel secure database model. In multilevel secure databases the main concern are mandatory access control, polyinstantiation and secure transaction processing, while in temporal databases the main concern is the addition of time support to record historical, present and future data. The temporal multilevel secure database model is a model that meets all of these concerns combined. In this study, we present an overview of the research and development efforts in both the area of temporal databases and multilevel secure databases, a definition of the new suggested model, the update operations that can take place in a temporal multilevel secure database and the integrity constraints that we need to meet in a temporal multilevel secure database in order to make sure that all the data inserted in the database is consistent, meaningful, historical and secure. A prototype of this model has been implemented. A detailed description and some of the example queries executed using this prototype are also presented in this study

More work is still to be done in this area. The temporal database used in this model is a validtime database; therefore the developed temporal multilevel secure database model was a validtime temporal multilevel secure model. In this model the only time attribute added to the model was the validtime interval. Much more interesting work remains to be done in the area of bitemporal multilevel secure database model, a model that will support the recording of both the transaction time and the validtime interval and the transaction-time temporal multilevel secure model, that will support only the recording of the transaction time.

REFERENCES

1. Department of Defense, 1985. National Computer Security Center. Department of Defense Trusted Computer System Evaluation Criteria. DOD 5200.28-STD
2. Jajodia, S., S. Sandhu Ravi and T. Blaustein Barbara. Toward a Multilevel Secure Relational Data Model. Information Security, pp: 460-492
3. Lunt, T.F., D.E. Denning, R.L. Schell, M. Heckman and W.M. Shockley, 1990. The Sea View Security Model. IEEE Trans. Software Engg., 16: 6.
4. Haigh, J.T., R.C. O'Brien and D.J. Thomsen, 1991. The LDV Secure Relational DBMS Model.
5. Smith, K. and M. Winslett, 1992. Entity Modeling in the MLS relational Model. In Proc. Intl. Conf. Very Large Databases (Vancouver, Canada, Aug. 1992), IEEE Computer Society Press, Los Alamitos, CA, pp: 199-210.
6. Mckenzie, E., 1986. Bibliography: Temporal databases. ACM SZGMOD Rec., 15: 40-52.
7. Ben-Zvi, J., 1982 The Time Relational Model. Ph.D. Thesis. Department of Computer Science, University of California, Los Angeles, CA.
8. Ariav, G., A. Beller and H.L. Morgan, 1984. *A* Temporal Model (Technical Report DS-WP 82-12-05). Department of Decision Sciences University of Pennsylvania, Philadelphia, PA. Bar-Hillel, Yehoshua 1954 Indexical Expressions. Mind, 63: 359-379.
9. Snodgrass, R., 1984. The Temporal Query Language TQUEL. In Proc. Third ACM SIGACT-SIGMOD Symp. Principles of Database Systems, Waterloo, Ontario, Canada.
10. Lum, V., *et al.*, 1984. Designing DBMS Support for the Temporal Dimension. In ACM–SIGMOD Intl. Conf. Management of Data. Boston, MA.
11. Clifford, J., 1985. Towards an Algebra of Historical Databases. In ACM-SIGMOD Intl. Conf. Management of Data. Austin, TX.
12. Snodgrass, R. and I. Ahn, 1985. A Taxonomy of Time in Data Bases. In ACM-SIGMOD Intl. Conf. Management of Data. Austin, TX.
13. Gadia, S.K. and J. Vaishnav, 1985. A query language of homogenous temporal data base. In Proc. Fourth Ann. ACM SIGACT-SIGMOD Symp. Priniciples of Database Systems.
14. Tansel, A., J. Clifford, S. Jajodia, A. Segev and R. Snodgrass, 1993. Temporal Databases: Theory, Design and Implementation. The Benjamin/Cummings Publishing Company, Redwood City.
15. Wu, Y., S. Jajodia and X. S. Wang. Temporal Database Bibliography Update, pp: 338-366.
16. Snodgrass, R.T., 2000. Developing Time-Oriented Database Applications in SQL. Morgan Kaufmann Publishers.
17. Shashi K. Gadia, 1998. Applicability of temporal data models to query multilevel security databases: A case study. Computer Science Department Iowa State University.