

An Authorization Model for Temporal and Derived Data: Securing Information Portals

Vijayalakshmi Atluri and Avigdor Gal
Rutgers University

The term *information portals* refers to Web sites that serve as main providers of focused information, gathered from distributed data sources. Gathering and disseminating information through information portals introduce new security challenges. In particular, the authorization specifications, as well as the granting process, are temporal by nature. Also, more often than not, the information provided by the portal is in fact derived from more than one backend data source. Therefore, any authorization model for information portals should support access control based on temporal characteristics of the data, and also should provide tools to prevent indirect unauthorized access through the use of derived data. In this paper we focus our attention on devising such an authorization model. The distinguishing features of this model include: (1) the specification of authorizations based on temporal characteristics of data, and (2) a formal framework to derive authorizations in a consistent and safe manner, based on relationships among data.

Categories and Subject Descriptors: C.2.4 [**Distributed Systems**]: Distributed Databases; M.2.0 [**General**]: Security, Integrity, and Protection

General Terms: Security

Additional Key Words and Phrases: Access Control, Authorization Administration, Derived Data, Temporal Data

The work of V. Atluri was supported in part by the National Science Foundation under grant IRI-9624222. A preliminary version of this work, consisting of an authorization model for temporal data only, was introduced by the authors in a paper presented at the 7th ACM Conference on Computer and Communication Security, 1-4 November 2000, Athens, Greece [Gal and Atluri 2000].

Address: V. Atluri, Department of Management Science and Information Systems, and Center for Information Management, Integration and Connectivity (CIMIC), Rutgers University, 180 University Avenue, Newark, NJ 07102 USA, phone: (973) 353-1642, fax: (973) 353-5808, email: atluri@andromeda.rutgers.edu; A. Gal, Department of Management Science and Information Systems, Piscataway, NJ 08854 USA, phone: (732) 445-3245, fax: (732) 445-6329, e-mail: avigal@rci.rutgers.edu, On leave from Rutgers University and serves as a Senior Lecturer at the Faculty of Industrial Engineering & Management at the Technion-Israel Institute of Technology.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or direct commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works, requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept, ACM Inc., 1515 Broadway, New York, NY 10036 USA, fax +1 (212) 869-0481, or permissions@acm.org.

1. INTRODUCTION

The term *information portals* refers to Web sites that serve as main providers of focused information (*e.g.*, financial information), gathered from distributed data sources. Just like general purpose portals (*e.g.*, Yahoo),¹ information portals serve as providers of value added information, yet their narrow scope allows them to build on the domain knowledge to optimize their information gathering and to maximize the quality of information they deliver. Information portals use backend databases as data providers and Web technology to format and present information. Typical applications that perform such tasks include portfolio management, digital libraries, and virtual retailing. For example, financial information providers (*e.g.*, Thomson Financial)² stream snapshots, as well as aggregated, financial information to consumers from various stock exchange markets. Also, digital libraries (*e.g.*, the Consumer's Report DL)³ provide conditional access to digital archives. Finally, eCommerce vendors (*e.g.*, Amazon.com)⁴ provide information such as availability and pricing of its products, gathered from numerous distributors.

Gathering and disseminating information through information portals introduce new security challenges. In particular, the authorization specifications, as well as the granting process, are temporal by nature. For example, the support of snapshots as well as aggregated information, requires tight management of the data streaming into and out of the database. Also, the borrowing process in a digital library calls for temporal control over the “borrowed” electronic copies. Finally, product information is transient and can be collected and aggregated for further analysis by the vendor. Another security challenge involves derived data, commonly used by information portals. Derived data is processed from backend data sources using *mappings*, the simplest of which would be a straightforward replication. Accessing derived data, as well as base data sources, requires a careful analysis of data inter-relationships to ensure consistent access granting. Otherwise, the provision of both information (*i.e.*, “processed data”) and raw data from related sources may (directly or indirectly) reveal prohibited data. Therefore, any authorization model for information portals should support access control based on temporal characteristics of the data, and should also provide tools to prevent indirect unauthorized access through the use of derived data.

In this paper, we propose an authorization model for information portals. In a nutshell, our model allows the specification of authorizations based on temporal characteristics of data, and provides a formal framework to derive authorizations in a consistent and safe manner, based on relationships among data. With respect to the temporal specification, access can be granted to a subject based on data validity or data update time, using either absolute or relative time references. As an example for the essential role of time in providing access control to data, consider an enterprise data warehouse where some data (possibly the price of an item) is computed predictively for a future time interval. Such data should not be made accessible to a salesperson before it becomes current. Thus, there is a

¹<http://www.yahoo.com>

²<http://thomsoninvest.net>

³<http://www.consumerreports.org>

⁴<http://www.amazon.com>

need to provide access control based on the valid time of the data. As another example, consider financial information service providers [Gal 1999], the providers of added-value information in financial applications. Such providers may segment the market by selling time sensitive data based on its currency. Such data may become public domain after a certain time delay, yet more recent data can be purchased. Therefore, to accommodate the access to privileged users, there is a need to impose access control based on data capture time. As a final example, a digital library should be able to enforce access control based on the time at which a document has been “checked-out,” (referred to in this paper as *replication time*) to allow flexible digital library policies for restricting access to a digital book after the expiry of the temporal-based authorization. A partial solution to this problem has been recently suggested by Authentica,⁵ whose product (PageVault) enables the user to select a specific time for a document to self destruct.

With respect to derived data, we define the concept of *safe authorization*, which enables the derivation of authorization for derived information based on the nature of mappings, and a subject’s accessibility to base data and mappings. More specifically, we classify mappings based on whether or not some base data can be reconstructed from them and the data they derive. We then ensure that authorization specification on the derived data should not derive non-existing authorization on the base data. As a simple example, consider an average salary attribute in a data warehouse. Such data may be considered less sensitive than the individual salaries from which it is derived. Our authorization model captures such policies by defining safe derivation of authorization. Therefore, we can allow access to the average salary attribute to a subject even if the subject does not have permissions to any individual salary since the mapping (*i.e.*, average) is of non-reconstructive type (*i.e.*, one cannot reconstruct individual salaries from the average salary value).

While existing authorization models provide adequate support for conventional databases, we are of the opinion that no authorization model can be found in the literature that is capable of providing access control to data that is both temporal and derived (see Section 8 for a thorough review of relevant research). Hence, the contribution of the paper is to develop an authorization model suitable of providing access control for such data. In particular, we provide a rule to derive safe authorizations and we formally show that any authorization base that uses safe authorizations is guaranteed to be consistent.

To demonstrate our approach, consider an international distributor with three branches, one of which is at the US and two more are at the European Union, located at Italy and Germany. Each branch maintains its own information regarding inventory and sales. Also, the basis for product pricing is the same for all branches, yet each branch has its own add-on elements (*e.g.*, Value Added Tax – VAT and sales tax). An information portal that utilizes the three databases provides retailers with information regarding the availability and pricing of products. Exchange rates are utilized in generating invoices, and therefore the databases maintain history of exchange rates. Throughout the paper we shall show how our authorization model supports temporal data (such as exchange rates and future pricing) and ensures consistent authorization base throughout the organization. In particular, we shall

⁵<http://www.authentica.com>

demonstrate the temporal component of the model, by providing a formula for computing foreign currency base pricing using the closing exchange rate of the previous day. Also, we shall show an example where inconsistent derivation of authorization allows representatives of one branch to access indirectly information of other branches, even though they are not legitimately allowed such access.

The rest of the paper is organized as follows. After introducing preliminary concepts in Section 2, we introduce in Section 3 a data model that naturally captures many of the characteristics of information as presented by information portals. An authorization model, which is tightly coupled with the data model and which extends existing authorization models to allow a refined (yet unified) access control for information portals is presented and discussed in sections 4 and 5. Access control within this framework is discussed in Section 6. Section 7 provides a discussion of design and implementation. The paper is concluded with a review of related work (Section 8) and a summary of our contributions (Section 9).

2. PRELIMINARIES

In this section we introduce some basic concepts that will assist us in outlining the principles behind our model. Section 2.1 introduces basic temporal concepts. Subjects and privileges are discussed in Section 2.2.

2.1 Temporal concepts

A *temporal domain* is a pair $(\mathcal{T}; \leq)$ where \mathcal{T} is a nonempty set and \leq is a total order on \mathcal{T} . Following previous works in the temporal database area, we adopt a discrete model of time [Clifford and Tansel 1985], isomorphic to the natural numbers ($\mathcal{T} \cong \mathbf{N}$). The model defines a *time point*⁶ $t \in \mathcal{T}$ to be a nondecomposable unit of time whose granularity is application-dependent. A *time interval* is designated as $\bar{t} = [t_s, t_e)$, the set of all time points $t \in \mathcal{T}$ such that $t_s \leq t < t_e$. Clearly, a time point $t \in \mathcal{T}$ can be represented as the time interval $[t, t + 1)$. For completeness, we define a time interval $\bar{t} = [t_s, t_e)$ to be *empty* if $t_s \geq t_e$. We define $\bar{\mathcal{T}}$, the set of all possible time intervals, to be the *time interval domain*.

Temporal database models associate one or more *temporal dimensions* with data, assigning different semantics to each dimension. For example, the temporal infrastructure document [Pissinou et al. 1994] advocates a bi-temporal database model, in which each datum is associated with two temporal dimensions, called valid time and transaction time. A *valid time* $(t_v = (t_s, t_e) \in \bar{\mathcal{T}}_v)$ is a time interval at which a datum is considered to be true in the modeled reality. A *transaction time* $(t_x \in \mathcal{T}_x)$, under a common interpretation, is a time point that designates the time in which the transaction that inserted the datum to the database was committed. Other temporal dimensions may be added, based on the application needs. For example, a digital library may attach a replication time $(t_r \in \mathcal{T}_r)$, reflecting the time at which an electronic copy of a book was replicated onto a borrower's device.

We end this section with the syntax of a temporal logic, to be used later in the paper. We use an instance of the temporal logic model suggested in [Shoham 1988]. We define TC to be a set of time point symbols, such as July30:1999:13:57

⁶While the term *chronon* was coined as a standard term in temporal databases [Jensen et al. 1992], we shall prefer the use of this more intuitive term.

and 5minutes, and $TV = \{t_x, t_s, t_e, t_r, t_{req}\}$ to be a set of temporal variables. As will become evident shortly (see Section 4 and Section 6), we need to timestamp access requests ($t_{req} \in TV$), in order to determine authorization granting. $TF = \{+, -, *, \div\}$ is a set of temporal function symbols, with a fixed arity of 2. We define R to be a set of relation symbols, each with a fixed arity of 1. Each relation shall correspond to a single property as will be explained shortly in Section 3. Also, we use O to define a set of object identifiers. Finally, we use C to define a set of non-temporal constants and V to define a set of non-temporal variables.

Definition 1. [Temporal terms] A temporal term trm is defined inductively, as follows:

- If $trm \in TC$, then trm is a temporal term;
- If $trm \in TV$, then trm is a temporal term;
- If trm_1 and trm_2 are temporal terms, and $f \in TF$ is a temporal function symbol, then $f(trm_1, trm_2)$ is a temporal term. \square

As examples of temporal terms, one may consider 5minutes, t_{req}, t_s, t_e and $+(t_x, 5minutes)$. For the latter, we shall use the conventional writing of $t_x + 5minutes$.

Definition 2. [Well-formed formulae] A well-formed formula (wff) is defined as follows:

- If trm_1 and trm_2 are temporal terms, then $trm_1 = trm_2$ and $trm_1 \leq trm_2$ are wffs;
- If trm is a non-temporal term, and $r \in R$ is a relation symbol, then $r(trm)$ is a wff;
- If φ and ψ are wffs, then $(\varphi \wedge \psi)$, and $(\neg\varphi)$ are wffs. \square

As an example, one may consider the wff $(t_s \leq t_{req} \wedge t_{req} \leq t_e)$.

2.2 Subjects and privileges

Let $S = \{s_1, s_2 \dots\}$ be a finite set of subjects, $M = \{read, write, own, \dots\}$ be a finite set of privilege modes and o a protected object to which an access may be either granted or denied. Subjects are categorized into a set of *privileged groups* (PG), e.g., preferred retailers versus other retailers. The factors that determine how subjects are grouped into privileged groups depend on the administrative policies of the application. A privileged group can therefore be a single subject or a group of subjects, where one subject can possibly participate in several groups. Moreover, the participation of a subject in a privileged group is dynamic. For example, it is not unusual for brokerage firms to provide a selected group of their customers with financial analysis based on the amount of transactions they perform within a given time frame. We assume that the elements of PG are organized into a lattice $\pi = (PG, E_\pi)$, with the following semantics. Let $\{pg_1, pg_2\} \subseteq PG$ such that there exists an edge $pg_1 \rightarrow pg_2 \in E_\pi$. Thus, any subject s who is in a privileged group pg_1 is also in a privileged group pg_2 . For example, given two privileged groups, *preferredRetailers* and *Retailers*, $preferredRetailers \rightarrow Retailers \in E_\pi$ means that any object o that is accessible to a retailer, is also accessible to a preferred retailer. We refer to *preferredRetailers* as being *higher* privileged group than

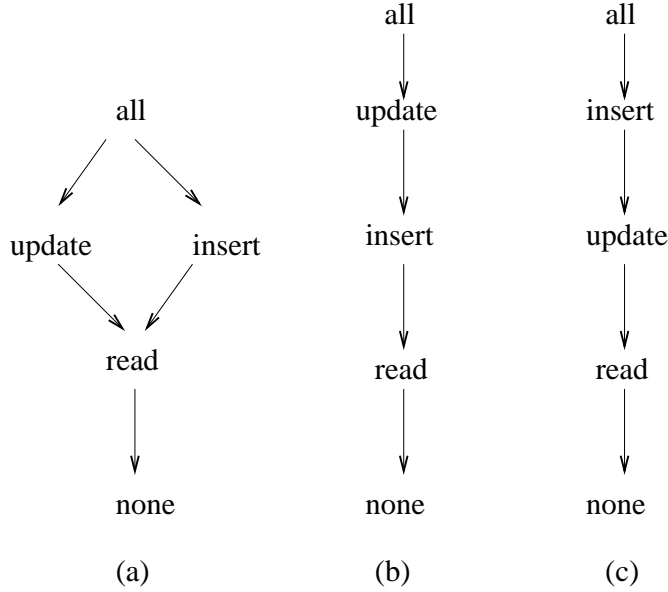


Fig. 1. Examples of Privilege Mode Hierarchies

Retailers. It is worth noting that the term privileged group resembles that of roles [Sandhu et al. 1996]. Yet, while roles are often related with organizational assignments, information portals support applications (and subjects) outside the realm of a single organization.

Similar to privileged groups, we assume that the set of privilege modes form a lattice $\mu = (M, E_\mu)$, with the following semantics. Let $\{m_1, m_2\} \subseteq M$ such that there exists an edge $m_1 \rightarrow m_2 \in E_\mu$. Thus, any subject s who can gain access to an object o in mode m_1 , can also gain access to o in mode m_2 . For example, given two privilege modes, *read* and *write*, then $write \rightarrow read \in E_\mu$ means that any subject s who can write to an object o can also read o . We refer to the *write* privilege mode as being higher than the *read* privilege mode.

The lattices π and μ set partial orders on privileged groups and privilege modes, respectively. Given two privileged groups pg_i and pg_j , $pg_i \preceq_\pi pg_j$ if either $pg_i = pg_j$ or there exists a path from pg_j to pg_i in π . Similarly, given two privilege modes m_i and m_j , $m_i \preceq_\mu m_j$ if either $m_i = m_j$ or there exists a path from m_j to m_i in μ . For example, based on the lattice in Figure 1(b) one may conclude that $read \preceq_\mu update$. For any lattice μ , there exists a greatest lower bound privilege mode (denoted **none**) such that $\forall m \in M, \text{none} \preceq_\mu m$, and a least upper bound (denoted **all**) such that $\forall m \in M, m \preceq_\mu \text{all}$.

The privilege hierarchy may vary among different organizations. For example, consider the three privilege mode lattices in Figure 1. In some cases, *update* and *insert* modes are incomparable as shown in Figure 1(a). In the case of derived data, the *update* privilege may be higher than the *insert* privilege in the privilege mode hierarchy (as shown in Figure 1(b)) since the owners of the base data, from

which the derived data is computed, may not wish the derived data to be updated by other subjects (giving them implicit access to the base data), yet may not object if the derived data is overridden by inserting a datum that is not derived from the base data. Alternatively, for some organizations, the *insert* privilege may be higher than the *update* privilege in the privilege mode hierarchy (as in Figure 1(c)). In what follows, we use $\text{lub}(m_1, \dots, m_n)$ to denote the least upper bound of privilege modes (m_1, \dots, m_n) . Similarly, we use $\text{glb}(m_1, \dots, m_n)$ to denote the greatest lower bound.

3. THE DATA MODEL

In this section we introduce a data model that captures the characteristics of information as provided by information portals. We start by introducing temporal data objects (Section 3.1), to be followed by definitions of mappings and derived data objects (Section 3.2).

3.1 Temporal data objects

Let $\bar{T} = \langle \bar{T}_1, \bar{T}_2, \dots, \bar{T}_n \rangle$ be a sequence of n time interval domains. Using object-based notation, $TS_{\bar{T}} = \{B_1, B_2, \dots, B_m\}$ is a temporal database schema that consists of m classes. A class B has p properties p_1, p_2, \dots, p_p . In relational database terms, B would be a relation and p would be an attribute.

*Example 1. [Database schema] Consider the provision of inventory information as provided by an information portal, using the inventory information available at the three local warehouses. The inventory information is partially represented using a class **Product** with the following five properties: **ProductNo**, **ProductName**, **Description**, **InventoryLevel**, and **PriceBase**. The class provides information regarding products and their aggregated inventory level. As mentioned earlier, the price base for all three branches is identical and is given in the information portal using \$US. □*

Each property has a *property domain* $\text{Dom}(p_j)$ ($1 \leq j \leq p$), and a *temporal property domain* $\text{TDom}(p_j) = \text{Dom}(p_j) \times \bar{T}_1 \times \bar{T}_2 \times \dots \times \bar{T}_n$ ($1 \leq j \leq p$), which is the Cartesian product of $n+1$ domains, one of which is the property domain, while the other n domains are time interval domains, referred to as the *temporal dimensions* of p_j . For example, a temporal property domain in a bi-temporal database can be represented as $\text{TDom}(p_j) = \text{Dom}(p_j) \times \bar{T}_v \times T_x$.

Definition 3. [Class instance] A class instance r of a class B is a sequence of sets $\langle s_1, s_2, \dots, s_{n_i} \rangle$, such that $\forall i$ ($1 \leq i \leq n_i$), $s_i \subset \text{TDom}(p_i)$. s_i ($1 \leq i \leq n$) is termed a state of property p_i and each element se in s_i ($1 \leq i \leq n$) is termed a state-element of property p_i . □

Definition 3 defines a class instance to be a sequence of sets, where the basic primitive (state-element) corresponds to a single value in a conventional database.⁷ A state s_i is the set of state-elements of the property p_i of the class instance c . As a notation convention, we overload the \in set symbol to identify the relationships among state-elements, states, properties, class instances and classes as follows. Let

⁷Class instances are commonly termed *objects*. In this paper, however, the term object is reserved for the authorization model (see Section 4).

PriceBase=	se_1 35, [July30:2000, September25:2000),	July30:2000:13:58
	se_2 30, [September25:2000, November30:2000),	July30:2000:13:58
	se_3 28, [<u>November30:2000, January2:2001</u>),	<u>October20:2000:15:56</u>
	t_v	t_x

Table 1. A PriceBase state example

se be a state-element of a state s , where s is a state of property p for a class instance r . p is a property of class B and r is an instance of B . Therefore, $se \in s$, $s \in p$, $s \in r$, $p \in B$, and $r \in B$. Also, we use r as a unique identifier of the class instance it represents. Finally, let se be a state-element of a state s with a temporal property domain $TDom(p)$. In further reference, we shall refer to the specific assignment of values to se as $se.val$ for the non-temporal value and $se.t_i$ ($1 \leq i \leq n$) for the associated temporal values.

Example 2. The temporal aspect of the **Product** class is transparent to the user, as can be readily seen from the schema description in Example 1. Table 1 illustrates the changes to the state **PriceBase** of the **Product** class for some class instance. The state represents the changes of the price base for the product over time.

The **PriceBase** for [September25:2000, November30:2000) is given in a prospective manner, since $t_x = \text{July30:2000:13:58}$ is earlier than the start of the valid time. \square

Equipped with the temporal data model, we are now ready to provide the semantics of wffs in this framework. We start by defining interpretation, to be followed by variable assignments.

Definition 4. [Interpretation] An interpretation is a quadruple $S = \langle TW, \leq, W, TFN \rangle$ where

TW is a nonempty universe of time points;

\leq is a binary relation on TW ;

W is a nonempty universe of individuals that is disjoint from TW ;

TFN is a set of total functions in $TW^2 \rightarrow TW$.

Let RL be a set of relations over W . M is a set of meaning functions where:

$$-M_1 : TC \rightarrow TW$$

$$-M_2 : C \rightarrow W$$

$$-M_3 : TF \rightarrow TFN$$

$$-M_4 : TW^4 \times O \times R \rightarrow RL \quad \square$$

Let TW be the integers. M_1 assigns a temporal time point symbol with an integer, using some application-dependent granularity (*e.g.*, minutes). M_2 provides an interpretation of the non-temporal constants, *e.g.*, subjects. M_3 provides the common interpretation for addition, subtraction, multiplication, and division in the integers. Finally, M_4 assigns a specific combination of temporal dimensions, an object identifier, and a relation with a value. M_4 returns a set, satisfying the (possibly partial) given conditions. M_4 is utilized in identifying state-elements that match certain temporal criteria. Some examples of the meaning functions M_1 and

M_4 are as follows:

$$\begin{aligned} M_1(\text{July30:2000:13:58}) &= 838, \\ M_1(\text{5minutes}) &= 5, \\ M_4(0, 83521, 838, \emptyset, r, \text{PriceBase}) &= \text{PriceBase}(35) \end{aligned}$$

In the first example, July30:2000:00:00 is taken as time 0. In the third example, the property `PriceBase` of an object with an object identity r , a transaction time of 838, a valid time of [0, 83521) (corresponding to [July30:2000, September:25:2000)), and no replication time, is mapped into a value 35 (based on Table 1).

For the purpose of variable assignments, we shall use five temporal assignments, as follows. In addition to the transaction time, t_x , and replication time, t_r , we shall represent each valid time interval $\bar{t}_v = [t_s, t_e)$ by its extreme time points and we shall use t_{req} as the time at which an access request was submitted to the database. Should additional temporal dimensions be needed, the variable assignment should include them as well. The formal definition of variable assignment follows closely that of [Shoham 1988].

Definition 5. [Variable assignment] A variable assignment is a pair $VA = \langle \{VAT\}^4, VAV \rangle$ such that $VAT : TV \rightarrow TW$ and $VAV : V \rightarrow W$. \square

Let PB be a non-temporal variable (representing the value of the `PriceBase` property of our sample product). An example of a variable assignment is

$$\begin{aligned} VA(t_x, t_s, t_e, t_r, PB) &= \langle VAT(t_x), VAT(t_s), VAT(t_e), VAT(t_r), VAV(PB) \rangle \\ &= \langle 838, 0, 83521, \emptyset, 35 \rangle \end{aligned} \quad (1)$$

Definition 6. [Time dependent meaning] A time dependent meaning MVA is defined inductively according to the following rules:

- If $tv \in TV$ then $MVA(tv) = VAT(tv)$.
- If $tc \in TC$ then $MVA(tc) = M_1(tc)$.
- If $f \in TF$ and $trm = f(trm_1, trm_2)$ is a temporal term then $MVA(trm) = M_3(f)(MVA(trm_1), MVA(trm_2))$. \square

3.2 Derived data objects

A *derived property* p is a property computed from properties p_1, p_2, \dots, p_p using a mapping $f(p_1, p_2, \dots, p_p)$. We refrain from introducing a specific language for mappings, and we shall use throughout the paper examples based on [Gal et al. 1996]. As an example, consider the property `InventoryLevel`, as presented in Example 1. This property is computed as a sum of the inventory levels of the three local warehouses, annotated as `InventoryLevel=US.InventoryLevel+GR.InventoryLevel+IT.InventoryLevel`, where `US`, `GR`, and `IT` represent the three databases from which the information portal derives its information. It is worth noting that the derivation language supports implicit join conditions (see, for example, [Etzion 1993]), where in this case, the three various inventory levels are matched through the unique `ProductNo` property.

Example 3. [Time varying mapping] We next introduce an example for time varying mappings. Consider the conversion from `US.PriceBase` to `IT.PriceBase`, using

an additional property `IT.US2ITConversion`, a time sensitive property. The conversion uses an exchange rate of \$US to ITL, taken at a given date, as a basis for the calculation of the price base for the following date. This derivation may be captured using the following mapping (based on [Gal et al. 1996]):

$$\begin{aligned} \langle \text{IT.PriceBase}, [\text{date}(t) + 1 \text{ day}, \text{date}(t) + 2 \text{ days}] \rangle = \\ \langle \text{IT.US2ITConversion}, [\text{date}(t), \text{date}(t) + 1 \text{ day}] \rangle \\ * \langle \text{US.PriceBase}, [\text{date}(t), \text{date}(t) + 1 \text{ day}] \rangle \end{aligned} \quad (2)$$

where `date(t)` stands for the date transformation of a time point variable, and the interval associated with `IT.PriceBase`, `IT.US2ITConversion`, and `US.PriceBase` represents the valid time dimension. The time point variable `t` serves in binding the valid time of any state-element of `IT.PriceBase` and `US.PriceBase` with a corresponding state-element of `IT.US2ITConversion`, to satisfy the conversion requirements as stated above. It is worth noting that without the temporal constraint, as given through the time point variable `t`, a default valid time is computed. In this case, the valid time of the computed `IT.PriceBase`'s state-element is computed to be the intersection of valid times of the state-elements of `IT.US2ITConversion` and `US.PriceBase` that participate in the computation. \square

We next extend the definition of a property to include derived properties. An extended property consists of information regarding its domain and the computation of its instances.

Definition 7. [Property] A property p of a class C is a pair $\langle TDom(p), f(P) \rangle$, such that $TDom(p)$ is the temporal domain of p , and f is a mapping.

The relationship between a property and its deriving properties ranges from *strictly defined* to *leniently defined*. If a relationship is strictly defined, a subject can only perform manipulations which are read-based (*e.g.*, read, select, etc.) on the derived data. In contrast, if the relationship is leniently defined, the property can be freely updated by the subject. An interesting case in this category is whenever there is a bidirectional update relationships between a property and its deriving properties, as in updateable replicas in a distributed databases. As another example to leniently defined relationship, consider a digital library with rewritable books. A book can be borrowed from the library, generating an updateable replication for a limited time period. The book is then updated and returned to the library at a later time (which is a de-facto update of the base data). The authorization model in Section 5 handles such a scenario by generating a safe assignment of access modes, thus synchronizing access control between base data and replications.

Authorizations on derived data are dependent on the nature of the mapping, and therefore we categorize the derivation mapping into two categories as follows:

(1) **Reconstructive type:**

Given a property p with the derivation mapping $f(P)$, f is said to be of reconstructive type if there exists an inverse $f^{-1}(p)$ and p is strictly defined.

As an example, assume that the property `GR.PriceBase` is computed from the property `IT.PriceBase` using the mapping `GR.PriceBase=989.999*IT.PriceBase` (note that the EU currencies are locked into a fixed exchange rate). Clearly,

PriceBase=		
se_1	35, [0,83521),	838
se_2	30, [83521, 177121),	838
se_3	28, $\underbrace{[177121, 224641)}_{t_v}$,	$\underbrace{119036}_{t_x}$

Table 2. A PriceBase state example, simplified

the inverse mapping exists ($IT.PriceBase=0.00101010*GR.PriceBase$) and thus the mapping is of a reconstructive type.

(2) Non-reconstructive type:

Given a property p with the derivation function $f(P)$, f is said to be of non-reconstructive type if there does not exist an inverse function $f^{-1}(p)$ or if p is leniently defined.

As an example, consider the conversion mapping, as introduced in Formula 2 above. Clearly, there is no inverse function for computing $IT.US2IT Conversion$ and $US.PriceBase$ from $IT.PriceBase$. In Section 5, however, we show that additional information may change a non-reconstructive mapping type into a reconstructive one. Such a mapping will be classified to be *partially reconstructive*.

4. TEMPORAL AUTHORIZATIONS

Let $O = \{o_1, o_2 \dots\}$ denote a set of authorization objects (objects for short). Authorization objects are those database elements for which authorizations can be specified and enforced. The granularity of an authorization object may correspond to either a class or a property. For simplicity sake, we shall assume in what follows that an authorization object corresponds to a single property. The model can be easily extended to either a set of properties or a class. The following definition allows specification of authorization based on the temporal attributes of objects.

Definition 8. [Authorization] An authorization \mathbf{a} is a quintuple $(pg, o, m, sign, \tau)$, where pg is a privileged group, o is an object, m is a privilege mode, $sign \in \{+, -\}$ indicating access or denial, and τ is a well-formed formula. \square

According to Definition 8, a subject s that belongs to a privileged group pg can gain (or be denied) a privilege m on state-elements of o , as determined by τ . τ is a well-formed formula that may involve one or more temporal dimensions of the object. Without the τ specification, either all state-elements of object o are accessible to a subject or none. The following examples demonstrate several situations where a more refined access control is needed, by using τ .

Example 4. [Authorization based on valid time] Consider our information portal case study. The various state-elements of PriceBase, as introduced in Table 1, are simplified for presentation purposes in Table 2 (using July30:2000:00:00 as time 0) by replacing each of t_s , t_e , and t_x with their M_1 interpretation.

Assume that there exist two privileged groups in the Italian branch, namely 1stLevelManager and 2ndLevelManager. An access control policy allows first level managers to access only the current Italian price base, but not predictive prices. In our model, this policy can be specified as follows: $(1stLevelManager, IT.PriceBase, read, (t_s \leq$

$t_{req} \wedge t_{req} \leq t_e$). Thus, if $t_{req} = 500$, only se_1 will be granted, since $0 \leq 500 \wedge 500 \leq 83521$.

Similarly, the policy that second level managers are allowed to update current, as well as up to one year ahead, US price base versions, is expressed as follows:

$$(2ndLevelManager, US.PriceBase, update, (t_s \leq t_{req} + 1year \wedge t_{req} \leq t_e) \quad (3)$$

In this case, if $t_{req} = 500$, access would be granted to create se_2 , since $t_{req} + 1year = 500 + 525600 = 526100$ and $83521 \leq 526100 \wedge 500 \leq 177121$.⁸ \square

Example 5. [Authorization based on transaction time] As another example, consider a stock exchange and assume that subjects belonging to a specific privileged group (say pg) are allowed access to a certain object (o), say the last trade size of various companies, only five minutes after it has been written into the database.

The authorization can be specified as follows: $(pg, o, read, +, t_x + 5minutes \leq t_{req})$. The wff $t_x + 5minutes \leq t_{req}$ establishes the 5 minutes delay security policy. For example, consider two state-elements of a specific company that is traded on the stock exchange, with transaction times $se_1.t_x = 58$, and $se_2.t_x = 64$. If $t_{req} = 63$, se_1 is selected, since $se_1.t_x = 58$ and thus $t_x + 5minutes = 58 + 5 = 63 \leq t_{req}$. Both se_1 and se_2 are selected for $t_{req} = 69$, (but not earlier since only at time 69 the assignment $se_2.t_x = 64$ is satisfied). It is worth noting that in the absence of any valid time constraint, a state-element will be presented to the user even if its valid time has already expired. For example, assuming $se_1.t_v = [57, 63)$, it remains accessible by a subject from the pg group even after time 63. \square

Example 6. [Authorization based on replication time] Consider a digital library setting, in which digital copies of books self-destruct 21 days from time of download. Each digital copy may be timestamped with a replication temporal dimension t_r . A general borrowing policy can thus be written as the authorization (Borrower, bookID, read, $t_r + 21days \leq t_{req}$). Therefore, 21 days after the time the book has been borrowed (t_r), the borrower will no longer be authorized to access it. \square

τ is utilized in identifying the state-elements for which an authorization may be granted, through the notion of formula satisfaction, as follows. Let S be an interpretation and VA a variable assignment. S satisfies a wff τ with VA (written as $S \models_{VA} \tau$) under the following inductively defined conditions:

- $S \models_{VA} trm_1 = trm_2$ iff $MVA(trm_1) = MVA(trm_2)$.
- $S \models_{VA} trm_1 \leq trm_2$ iff $MVA(trm_1) \leq MVA(trm_2)$.
- $S \models_{VA} r(trm)$ iff $r(VAV(trm)) \in M_4(VAT(t_x), VAT(t_s), VAT(t_e), VAT(t_r), r)$.
- $S \models_{VA} (\tau_1 \wedge \tau_2)$ iff $S \models_{VA} \tau_1$ and $S \models_{VA} \tau_2$.
- $S \models_{VA} (\neg \tau_1)$ iff $S \not\models_{VA} \tau_1$.

Example 7. Let S be an interpretation and let $VA = \langle 0, 83521, 838, \emptyset, 35 \rangle$ be the assignment, as introduced in Formula 1. Finally, let

$$\tau = (t_s \leq t_{req} \wedge t_{req} \leq t_e)$$

be a wff. We shall now prove that $S \models_{VA} \tau$, assuming that $t_{req} = July30:2000:08:20$.

⁸There are 525600 minutes in a year.

$-\tau = \tau_1 \wedge \tau_2$, where $\tau_1 = t_s \leq t_{req}$ and $\tau_2 = t_{req} \leq t_e$.

$-S \models_{VA} \tau_1$ since

$$\begin{aligned} MVA(t_s) &= M_1(t_s) = 0 \\ MVA(t_{req}) &= M_1(t_{req}) = 500 \end{aligned}$$

and

$$0 = MVA(t_s) \leq MVA(t_{req}) = 500.$$

$-S \models_{VA} \tau_2$ since

$$\begin{aligned} MVA(t_{req}) &= M_1(t_{req}) = 500, \\ MVA(t_e) &= M_1(t_e) = 83521 \end{aligned}$$

and

$$500 = MVA(t_{req}) \leq MVA(t_e) = 83521.$$

—Finally, $S \models_{VA} \tau$ since $S \models_{VA} \tau_1$ and $S \models_{VA} \tau_2$. □

A subject can be granted access to a state-element se under $\mathbf{a} = (pg, o, m, sign, \tau)$ only if $S \models_{VA(se.t_x, se.t_s, se.t_e, t_{req}, se.val)} \tau$. Therefore, Definition 8 refines conventional authorization specifications in that it may restrict the access to some state-elements of the authorized object, based on τ . In what follows we shall use the convention in which the absence of a temporal wff is interpreted as putting no restrictions on the temporal dimensions of state-elements. Such interpretation is easily achieved by assigning τ with a tautology (e.g., $t_s \leq t_s$).

It is also possible that there exist security policies that need to have authorizations specified on the mapping itself. For example, a broker may provide market forecast for selected customers. While the forecast itself may not be confidential (at least it can be sold or shared with a selective set of users), the formulae that are utilized in deriving the forecast are typically confidential. Therefore, there is a need to have access control on the mapping itself. The following rule authorization allows one to specify such access control policies.

Definition 9. [Rule authorization] A rule authorization \mathbf{ra} is a quintuple $(pg, f(P), m, sign, \tau)$, where pg is a privileged group, $f(P)$ is mapping, m is the privilege mode, $sign \in \{+, -\}$ indicating access or denial, and τ is a wff. □

In what follows, we use $pg_a, o_a, f_a, m_a, sign_a$, and τ_a to denote a privileged group, an object, a mapping, a privilege mode, access or denial, and a constraining formula of \mathbf{a} , respectively. The set of all authorization specifications is termed the *authorization base AB*.

4.1 Authorization unification

For all practical purposes, it becomes more efficient to provide access control using a unified authorization base, where authorizations that differ solely in their temporal formula are joined together. Such authorizations are unified into a single authorization with a modified temporal formula, as follows. Let $\{\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n\} \in AB$ be n authorizations, such that for any i and j ($1 \leq i < j \leq n$), $pg_{a_i} = pg_{a_j}$, $o_{a_i} =$

$o_{a_j}, m_{a_i} = m_{a_j}$, and $sign_{a_i} = sign_{a_j}$. A unification of $\{\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n\}$ is a process in which $\{\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n\}$ are replaced with a single authorization

$$\mathbf{a} = (pg_{a_1}, o_{a_1}, m_{a_1}, sign_{a_1}, \bigvee_{1 \leq i \leq n} \tau_i)$$

where $\bigvee_{1 \leq i \leq n} \tau_i$ denotes the disjunction of all temporal wffs. Given an authorization base \overline{AB} , unification of all authorizations in \overline{AB} can be performed in $O(n \log n)$, where n represents the number of authorizations in \overline{AB} , which is the cost of sorting the authorizations in \overline{AB} . We term an authorization base in which all possible unifications have been performed a *unified authorization base* UAB .

5. CONSISTENT AUTHORIZATIONS

Information portals utilize data from numerous data sources in deriving information. In this section, we provide a methodology for ensuring a consistent set of authorizations, taking into account the authorization set on the base data as well as the authorization set on the information at the information portal. The owners of the base data need to ensure that both confidentiality and integrity of the base data are preserved as per their security policies. That is, the owner of the base data does not wish to reveal the data to unintended audiences, neither does the owner wish her data be transformed by unauthorized people. As an example, consider the authorizations as given in Example 4 above, where first level managers are authorized to read the current price base in the IT database and second level managers are allowed to update the price base in the US database. John, who is not a manager, is responsible for updating the database with the daily \$US to ITL conversion rate. Therefore, he has gained the following authorization ($\text{John}, \text{IT.US2ITConversion}, \text{write}, (t_s \leq t_{req} \wedge t_{req} \leq t_e)$). To enable an automatic derivation of IT.PriceBase in response to the update of $\text{IT.US2ITConversion}$, John is also given the following two authorizations:

$$(\text{John}, f(\text{US.PriceBase}, \text{US2ITConversion}), \text{execute}, +, (t_s \leq t_{req} \wedge t_{req} \leq t_e)) \quad (4)$$

$$(\text{John}, \text{IT.PriceBase}, \text{write}, +, (t_s \leq t_{req} \wedge t_{req} \leq t_e)) \quad (5)$$

where f is the mapping depicted in Formula 2. Similarly to authorizations on views in conventional databases, John needs an authorization to perform the conversion of the US price base to the Italian price base. Without it, the automatic propagation of the new PriceBase would fail. Also, Authorization 5 enables the writing of the outcome into the Italian database. Assume further that the organization's access mode lattice includes the following edges, $\text{execute} \rightarrow \text{write}$ and $\text{write} \rightarrow \text{read}$. From the transitivity of the lattice, John gains a *read* access to Formula 2 (which, by itself, is a violation of the security policy defined in Example 4 since John is not a manager), and by the $\text{write} \rightarrow \text{read}$ rule, John also gains a *read* access to IT.PriceBase . John can now deduce US.PriceBase , using the mapping

$$\langle \text{US.PriceBase}, [\text{date}(t), \text{date}(t) + 1\text{day}] \rangle = \frac{\langle \text{IT.PriceBase}, [\text{date}(t) + 1\text{day}, \text{date}(t) + 2\text{days}] \rangle}{\langle \text{IT.US2ITConversion}, [\text{date}(t), \text{date}(t) + 1\text{day}] \rangle}$$

Changing temporal variables, one has that

$$\langle \text{US.PriceBase}, [\text{date}(t) - 1\text{day}, \text{date}(t)] \rangle = \frac{\langle \text{IT.PriceBase}, [\text{date}(t), \text{date}(t) + 1\text{day}] \rangle}{\langle \text{IT.US2ITConversion}, [\text{date}(t) - 1\text{day}, \text{date}(t)] \rangle} \quad (6)$$

and therefore John gains a read access to US.PriceBase (although with a one day delay), which he was unable to gain otherwise, not being a manager.

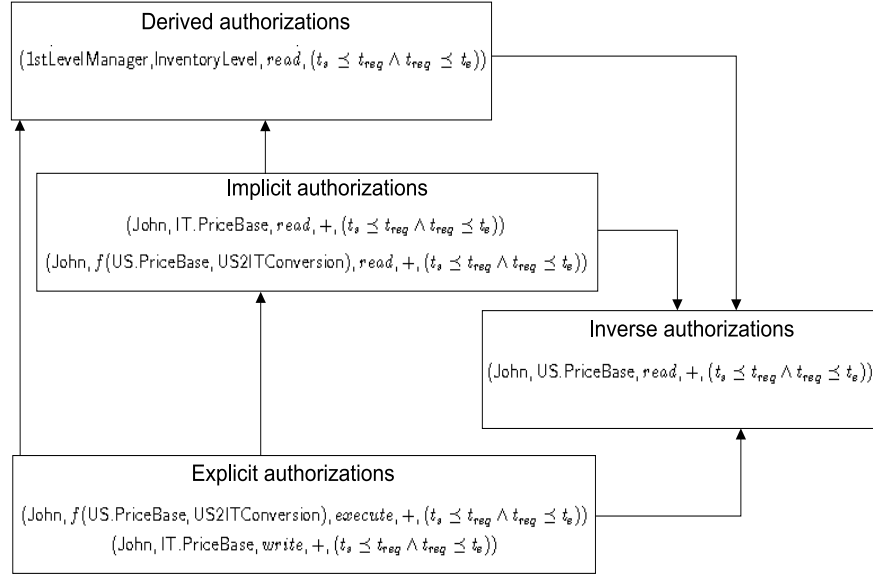


Fig. 2. Authorization types

The rest of this section shall formalize the method of avoiding such security risks, as depicted above. To accomplish this, we first identify four types of authorizations – explicit, implicit, derived, and inverse (Section 5.1). We then define authorization base consistency and provide a sufficient condition for consistency (Section 5.2).

5.1 Authorization types

This section provides four types of authorizations. The categorization assists us in achieving authorization consistency, as will be demonstrated in Section 5.2. Figure 2 provides a pictorial summary of the various authorization types and their relationships.

Explicit authorization. These authorizations can be specified on base data, derived data, and the mappings used in computing the derived data. As an example of an explicit authorization on base data, one may consider Authorization 3 given in Example 4 above. Also, Authorization 4 is an example of an authorization for mapping and Authorization 5 is an example of an authorization for derived data. The set of all explicit authorizations constitute the *authorization base AB*. As detailed in Section 4.1, *AB* can be transformed into a *unified authorization base UAB*, by unifying all authorizations that share all but the wff.

Implicit Authorization. These authorizations are implied through the privilege hierarchies. That is, if a subject possesses an authorization on an object with certain privilege mode, then all positive authorizations on that object that are lower than that privilege mode are implied as well as all negative authorizations of higher privilege modes. Also, if a subject belongs to a certain privileged group, then all positive authorizations of lower privileged groups will apply to her, as well as all negative authorizations of higher privileged groups. For example, Authorization 5

and the edge $write \rightarrow read \in E_\mu$ implicitly support the authorization

$$(\text{John}, \text{IT.PriceBase}, \text{read}, +, (t_s \leq t_{req} \wedge t_{req} \leq t_e)). \quad (7)$$

As another example, the authorization

$$(\text{John}, f(\text{US.PriceBase}, \text{US2IT Conversion}), \text{read}, +, (t_s \leq t_{req} \wedge t_{req} \leq t_e)) \quad (8)$$

is implicitly derived from Authorization 4 using the edges $\{execute \rightarrow write, write \rightarrow read\} \in E_\mu$. Section 5.1.1 provides a simple algorithm for identifying all implicit authorizations in a given UAB . We denote the set of all explicit and implicit authorizations the *expanded authorization base* EAB .

Derived authorization. Instead of specifying authorizations explicitly on the derived data, authorizations may be derived from the explicit (and implicit) authorizations specified on the participating authorization objects. Such authorization is termed a *derived authorization*, da , and the set of all derived authorizations is called *derived authorization base*, DAB . Given an object o , the set of all the derived authorizations in which o is involved is denoted by $DA(o)$. While there can be many authorization derivation methods, in Section 5.1.2 we shall introduce one specific method, which generates *safe authorizations*. We shall show in Section 5.2 that safe authorizations can be utilized in generating a sufficient condition for authorization consistency.

Inverse authorization. These authorizations can be deduced from a given set of explicit, implicit, and derived authorizations, and by applying the information regarding reconstructive mappings. That is, if a subject possesses authorizations on a certain derived object and on a subset of base data used to compute it, then authorizations on the remaining subset of the base data may be deduced. As an example of an inverse authorization, one may consider the authorization

$$(\text{John}, \text{US.PriceBase}, \text{read}, +, (t_s + 1\text{day} \leq t_{req} \wedge t_{req} \leq t_e + 1\text{day})),$$

deduced from the implicit authorizations 7 and 8. Section 5.1.3 formalizes the notion of inverse authorizations.

5.1.1 *Implicit authorization.* The introduction of the lattices μ and π in combination with AB generates implicit authorizations. In our model, implicit authorizations also have impact on authorizations temporal formulae. By way of motivation, consider a privilege hierarchy μ with $write \rightarrow read \in E_\mu$. Let \mathbf{a}_1 and \mathbf{a}_2 be the following two authorizations:

$$\begin{aligned} \mathbf{a}_1 &= (pg, o, \text{read}, +, t_x + 10\text{minutes} \leq t_{req}) \\ \mathbf{a}_2 &= (pg, o, \text{write}, +, t_x + 5\text{minutes} \leq t_{req}) \end{aligned}$$

The temporal implication of \mathbf{a}_2 on \mathbf{a}_1 is that

$$\tau_{\mathbf{a}_1} \leftarrow (\tau_{\mathbf{a}_1} \vee \tau_{\mathbf{a}_2}) \quad (9)$$

$$\begin{aligned} &= (t_x + 10\text{minutes} \leq t_{req} \vee t_x + 5\text{minutes} \leq t_{req}) \\ &= t_x + 5\text{minutes} \leq t_{req} \end{aligned} \quad (10)$$

where Formula 9 is derived from the semantics of the $write \rightarrow read$ relationship, and Formula 10 can be derived since $S \models_{VA} t_x + 5\text{minutes} \leq t_{req}$ entails $S \models_{VA}$

```

Expand/m/+:
Input: (pg, o, m, +, τ)
Output: (pg, o, m, +, τ')
Procedure:
  τ' := ∅
  If there exists a (pg, o, m, +, τ*) ∈ UAB then:
    τ' := τ*
  For each m' such that (m', m) ∈ Eμ do:
    (pg, o, m', +, τ'') := Expand((pg, o, m', +, τ))
    If τ'' ≠ ∅ then:
      τ' := τ' ∨ τ''
  end {For}
  If there exists a (pg, o, m, +, τ*) ∈ UAB do:
    UAB := UAB - (pg, o, m, +, τ*)
    EAB := EAB + (pg, o, m, +, τ')
  return (pg, o, m, +, τ')

```

Fig. 3. Formulae Expansion/m/+

$t_x + 10\text{minutes} \leq t_{req}$. In this example, although a *read* authorization is explicitly given for 10 minutes old data, the write authorization extends it to include data that is newer than 10 minutes yet not newer than 5 minutes, based on E_μ .

We next provide an approach for generating an *expanded authorization base* EAB from a unified authorization base UAB , in which for any authorization \mathbf{a} (either existing or newly generated), τ_a is modified to take into account the semantics of the privileged group and privilege mode lattices. The basic routine in this algorithm is the *Expand* recursive routine, which has four versions, namely, *Expand/m/+*, *Expand/m/-*, *Expand/pg/+*, and *Expand/pg/-*. Each of these versions handles a different combination of a hierarchy and a sign. *Expand/m/+* is given in Figure 3, and uses Depth-First-Search (DFS) to scan the m lattice. At each iteration, a possible authorization is given as an input for the algorithm. The algorithm is recursively called for each preceding node in μ . The returned temporal constraints are then combined and returned to the calling routine. The stop condition would be a root node in μ . It is worth noting that the expanded authorization base consists of modified existing authorizations as well as newly introduced authorizations. Therefore, in the example given above, τ_{a_1} will be modified into $(t_x + 10\text{minutes} \leq t_{req} \vee t_x + 5\text{minutes} \leq t_{req})$ (which is effectively $t_x + 5\text{minutes} \leq t_{req}$). Even had we had only \mathbf{a}_2 , a new authorization, $(pg, o, read, +, t_x + 5\text{minutes} \leq t_{req})$, would have been created in EAB .

The routine is performed for each authorization in UAB . Given a lattice μ with n nodes and a unified authorization base with m authorizations, computing EAB can be completed in a worse time complexity of $O(n \times m)$. To optimize the process, authorizations should be evaluated from the leaf nodes up, ensuring a maximum number of deleted authorizations from UAB during the process.

All four routines use DFS to scan the relevant lattice. The main difference between the presented algorithm and the routine that handles the privileged group hierarchy and negative authorizations lies in the ordering of the recursive calls. While in Figure 3 nodes are being scanned up the lattice, where each lower mode

contributes an additional disjunctive term to τ , in *Expand/m/-* nodes are being scanned down the lattice. The differences among the four routines are minor, and we refrain from providing the pseudo-code for *Expand/m/-*, *Expand/pg/+*, and *Expand/pg/-* in this paper.

Both positive and negative authorizations for the same combination of privileged group, object, and access mode (with possibly different formulae) may coexist in *EAB*. In such a case, we take a conservative approach and assume that negative authorizations prevail whenever overlapping intervals are concerned. Unlike conventional authorization models, a careful analysis of the temporal expressions in both authorizations is required to identify conflicts. For example, consider the following two authorizations:

$$\mathbf{a}_1 = (pg, o, write, +, t_x + 5\text{minutes} \leq t_{req})$$

$$\mathbf{a}_2 = (pg, o, write, -, t_x + 9\text{minutes} \leq t_{req})$$

A careful analysis concludes that the temporal expression of τ_{a_1} is further restricted by τ_{a_2} to be $(t_x + 5\text{minutes} \leq t_{req} \wedge \neg(t_{req} \leq t_x + 9\text{minutes}))$. For example, given the state-element se_1 from Example 5 above with $t_x = 58$, $t_{req} = 63$, and an interval access request for $[t_{req}, t_{req} + 10)$, \mathbf{a}_1 would allow access to se_1 during $[63, 73)$ and \mathbf{a}_2 would prevent access to se_1 during $[67, 73)$ (in both cases, 73 is taken from the access request terminating time point $t_{req} + 10\text{minutes} = 63 + 10 = 73$). The combined impact of both authorizations would be an access granting during $[63, 67)$.

5.1.2 Derived authorization. Authorizations of derived objects can either be explicitly specified by the security administrator of the derived data, or can be derived from the authorizations of the participating objects. The latter can provide guarantees on the security and privacy of the base data, provided the derivation rules are consistent with the access permissions on the base data. Towards this end, we assume that administrators of derived data have to comply with security policies of those of base data.

Let o be a derived object, derived using $f(P)$ from a set of participating base data objects P . If o is strictly defined, we are concerned that a subject s , which is not allowed to access $o_i \in P$ in a read-based privilege mode m , would be able to compute it from derived data, possibly enhanced with other information (*e.g.*, the mapping formula and other base data). For example, we have shown earlier that John was able to gain a de-facto read authorization on *US.BasePrice*, by accessing *IT.BasePrice* and *US2ITConversion* and using the inverse function of the mapping of *IT.BasePrice*. Therefore, to derive authorizations of the derived objects from those of the base objects, we need to decide on the type of the mapping. In addition to the reconstructive and non-reconstructive types, defined in Section 3.2, one must also consider a third type, namely a *partial reconstructive type*, which can be defined as follows. Given an object o with the derivation mapping $f(P)$, f is said to be of *partial reconstructive type* with respect to a privileged group pg if o is strictly defined and $f^{-1}(o)$ does not exist, yet there exist explicit authorizations for pg on $SUB_P \subset P$ such that $f^{-1}(o, SUB_P)$ exists. In the example given above, the mapping is of partial reconstructive type with respect to John, since an inverse mapping exists (Formula 6) with $SUB_P = \{\text{IT.PriceBase}\}$.

We next provide a method for deriving authorizations that ensure authorization

consistency, as will be discussed in Section 5.2. We assume authorization on derived objects are simply derived from those on base objects. Such authorization is denoted a *derived authorization*, da . The set of all derived authorizations is called *derived authorization base*, DAB . Given an object o , the set of all the derived authorizations in which o is involved is denoted by $DA(o)$.

Given an object o , the set of all the explicit authorizations in which o is involved is denoted by $EAB(o)$. Given a derived object $o(f(P))$, where $P = p_1, p_2 \dots p_n$, we define *relevant authorizations* of o , $rel(o)$ as follows: $rel(o) = EAB(p_1) \cup EAB(p_2) \cup \dots EAB(p_n)$.

Definition 10. [Authorizations of a subject] Given a set of authorizations EAB and a subject s_i , we define a subset of authorizations called authorizations of s_i , $EAB_{s_i} = \{ea \in EAB \mid s_{ea} = s_i\}$. The authorizations of s_i , EAB_{s_i} is the union of two disjoint sets $EAB_{s_i}^+$ and $EAB_{s_i}^-$ where $EAB_{s_i}^+ = \{ea \in EAB_{s_i} \mid sign_{ea} = +\}$ and $EAB_{s_i}^- = \{ea \in EAB_{s_i} \mid sign_{ea} = -\}$.

In the above definition, we partition authorization specifications for each subject, and the set of authorizations of each subject is further partitioned based on positive and negative authorizations.

In what follows, we use $m_{pg,o}^+$ to specify the set of all privilege modes to which privileged group pg is entitled in accessing object o . The default value of $m_{pg,o}^+$ is *null*. Similarly, we use $m_{pg,o}^-$ to specify the set of all privilege modes to which privileged group pg is not entitled in accessing object o . The default value of $m_{pg,o}^-$ is the set of all privilege modes.

Definition 11. [Safe authorization] Given a derived object o and a set of objects $P = \{o_1, o_2, \dots o_n\}$, an authorization $\mathbf{a} = (pg, o, m, sign, \tau)$ is safe with respect to P if one of the following two conditions hold:

- $sign = +$: the following conditions jointly hold:
 - $-m \preceq_{\mu} lub(\cup_{o' \in P} m_{pg,o'}^+)$
 - $-\sim \tau \vee (\bigwedge_{a \in rel(o)^+} \tau_a)$ is a tautology
- $sign = -$: the following conditions jointly hold:
 - $-glb(\cup_{o' \in P} m_{pg,o'}^-) \preceq_{\mu} m$
 - $-\tau \vee \sim (\bigvee_{a \in rel(o)^-} \tau_a)$ is a tautology □

Given a set of relevant explicit authorizations for a derived object, the above definition can be used to derive a derived authorization that is safe. The wff of a safe authorization should not hold whenever the conjunction of all the wffs of the authorizations of the participating objects does not hold (in case of positive authorizations) and the wff of a safe authorization must hold whenever the disjunction of the wffs of the participating objects hold (in case of negative authorizations).

Given P , a subset of the objects that participate in computing the derived object o , and a privileged group, the above definition can be used to derive a derived authorization that is safe. The following definition provides a mechanism for ensuring that any authorization for a derived object is safe. We assume that all authorizations are derived from the authorizations on the objects that participate in computing the derived object. As a notation convention, we use

$A(o, pg) = \{a \in EAB \mid o_a = o \wedge pg_a = pg\}$ to specify the set of authorizations of o for a privilege group pg .

Definition 12. [Safe derived authorization rule] A safe derived authorization rule for a derived object o (derived using a mapping $f(P)$) and a privileged group pg is given as follows:

- (1) if $(pg, f(P), read, +, \tau) \notin EAB$, or if $(pg, f(P), read, -, \tau) \in EAB$, or if f is of non-reconstructive type, then for every $a \in A(o, pg)$, such that $\sim \tau_a \vee \tau$ is a tautology, a is safe.
- (2) if f is of reconstructive type, then for every $a \in A(o, pg)$, a needs to be safe with respect to P .
- (3) if f is of partial reconstructive type with respect to pg and there is a set SUB_P such that $f^{-1}(o, SUB_P)$ exists, then for every $a \in A(o, pg)$ a needs to be safe with respect to $P - SUB_P$. \square

Definition 12 is utilized in Section 5.2 to guarantee a consistent authorization base. The first condition defines any authorization as safe, given that the mapping is either of non-reconstructive type or that a subject cannot gain read access to it. The $\sim \tau_a \vee \tau$ formula restricts this statement only to the times that read access is denied (or not allowed). It is worth noting that in this case, the authorization specifications need not be derived from the base data, rather it can be decided independently by the administrator of the derived database, *i.e.*, they are specified explicitly. The second condition states that the authorization specification of the derived object must be safe with respect to the whole set of deriving objects if the function utilized to derive it is of reconstructive type and the third condition states that if there exists a privileged group with authorizations on a subset of participating objects such that the function is of partial reconstructive type, then the authorization specification of the derived object must be safe with respect to that subset. It is also worth noting that the rule refers to all authorizations in the extended authorization base, *i.e.*, both explicit and implicit authorizations.

Example 8. [Safe authorizations] Consider the property `InventoryLevel`, as presented in Example 1. This property is computed as a sum of the inventory levels of the three local warehouses, annotated as `InventoryLevel = US.InventoryLevel + GR.InventoryLevel + IT.InventoryLevel`. For each of the following assumptions, we shall compute a safe authorization for `InventoryLevel`, or show that such an authorization does not exist.

- (1) The privileged group `retailers` does not have any read authorization to the mapping itself. Therefore, according to the first condition in Definition 12, the authorization $(retailers, InventoryLevel, read, +, (t_s \leq t_{req} \wedge t_{req} \leq t_e))$ is safe.
- (2) The privileged group `IT1stLevelManager` has the following authorization

$$(IT1stLevelManager, IT.InventoryLevel, read, +, (t_s \leq t_{req} \wedge t_{req} \leq t_e))$$

and no authorizations on either `US.InventoryLevel` or `GR.InventoryLevel`. Although Italian first level managers have read authorization on the mapping itself, the mapping is not of partial reconstructive type, since $SUB_P = \{IT.InventoryLevel\}$ and $f^{-1}(InventoryLevel, IT.InventoryLevel)$ does not exist.

Therefore, the authorization $(1stLevelManager, InventoryLevel, read, (t_s \leq t_{req} \wedge t_{req} \leq t_e))$ is safe, according to the first condition in Definition 12.

(3) Consider the authorizations given to John earlier:

$(John, IT.US2ITConversion, write, +, (t_s \leq t_{req} \wedge t_{req} \leq t_e))$

$(John, f(US.PriceBase, US2ITConversion), execute, +, (t_s \leq t_{req} \wedge t_{req} \leq t_e))$

The function is of partial reconstructive type, with $SUB_P = \{IT.US2ITConversion\}$.

Therefore, any authorization should be safe with respect to US.PriceBase. However, since John has no authorization for US.PriceBase, any safe authorization would have $m = \text{none}$. Thus, Authorization 5 is not safe. \square

It is worth noting that in the case of aggregated data, the explicit authorization for a derived object may be either higher or lower than the authorizations on the base objects. For example, a privileged group pg may be given read access to average salaries of all employees of an organization, but not allowed read access to any individual's salary. On the other hand, the total number of guests in a hotel may not be revealed to a privileged group pg , although access to a particular guest that is staying at the hotel may be granted. If the total number of guests is a derived object, appropriate authorizations may be assigned explicitly. However, it is important to note that independent of the authorizations on these derived data objects, the confidentiality of the base data is not compromised since the functions are of non-reconstructive type.

It is also worth noting that the definition of safe authorizations goes beyond the read access mode. For example, consider a derived database with an object o , where its relationships with its participants is leniently defined. If a subject does not possess a *write* privilege to the participants, yet she obtains a *write* authorization to o , a malicious subject may use this authorization to contaminate the derived data. Therefore, safe authorizations ensure the most conservative authorization assignment which, in the case of leniently defined data, does not allow a subject to gain more access to derived data than she already has for the base data.

5.1.3 Inverse authorization. Due to the reconstructive nature of the derivation mappings, subjects who have access to a derived object may be able to compute the base data even though they do not have access permissions on them. For example, John has gained a de-facto *read* permission on US.PriceBase. Such a permission is termed inverse authorization and is defined herein.

Definition 13. [Inverse authorization] Let $ra = (pg_{ra}, f_{ra}(P), m_{ra}, +, \tau_{ra})$ be a rule authorization, which derives o (where $P = \{o_1, o_2, \dots, o_n\}$), and let $a = (pg_a, o_a, m_a, +, \tau_a)$ be an authorization such that $o_a = o$ and $read \leq_{\mu} m_a$.

- (1) If f is of reconstructive type, then there exist inverse authorizations $(pg, o_j, m_j, +, (\tau_{ra} \wedge \tau_a))$ for all $m_j \leq_{\mu} read$ and $o_j \in P$, or
- (2) if f is of partial reconstructive type with respect to pg where $f^{-1}(o, SUB_P)$ exists, then there exist inverse authorizations $(pg, o_j, m_i, +, (\tau_{ra} \wedge \tau_a))$ for all $m_i \leq_{\mu} read$ and $o_j \in \{P - SUB_P\}$. \square

It is worth noting that inverse authorizations exist de-facto once the conditions in Definition 13 are satisfied. This is not to say that every inverse authorization should

be added to the authorization base. On the contrary, once an inverse authorization is identified, the authorization base should be modified to avoid illegitimate access to data.

Example 9. [Inverse authorization] Considering the authorizations of John as given above, one concludes that the inverse authorization $(\text{John}, \text{US.PriceBase}, \text{read}, (t_s + 1\text{day} \leq t_{req} \wedge t_{req} \leq t_e + 1\text{day}))$ exists, since the mapping is of partial reconstructive type, and Authorization 4 for the mapping as well as Authorization 5 for writing IT.PriceBase both exist. \square

The set of all inverse authorizations, as derived from an expanded authorization base EAB , is called the *inverse authorization base* of EAB , and is denoted $IAB(EAB)$.

In summary, AB is the set of authorizations explicitly specified, UAB is the set of authorizations after performing the unification on AB , EAB is the set of authorizations after expanding UAB to include implicit authorizations, DAB is the set of derived authorizations, and $IAB(EAB)$ is the set of inverse authorizations inferred from EAB . Section 4.1 provides an algorithm for computing UAB from AB in an $O(n \log n)$ time complexity, where n is the number of authorizations in AB . Section 5.1.1 provides an algorithm for computing EAB from UAB in a time complexity of $O(m \log n)$ time complexity, where $m = \max(|M|, |PG|)$ and n is the number of authorizations in EAB . Using Definition 11 and Definition 12, one can construct DAB in a linear time, where for each derived object, Definition 11 is applied with respect to the deriving objects in $P - SUB_P$. Finally, we are not concerned with the complexity of constructing $IAB(EAB)$. As will become evident shortly, the use of safe authorizations ensures that $IAB(EAB)$ does not contain unintended authorizations.

5.2 Authorization consistency

As mentioned earlier, the authorizations on derived data must preserve the confidentiality and integrity of the base data. In particular, all inverse authorizations must be either explicitly specified or safely derived. Our notion of authorization consistency, defined herein, reflects this requirement.

Definition 14. [Authorization consistency] An expanded authorization base EAB enjoys authorization consistency if for any authorization $a' \in IAB(EAB)$ there exists an authorization $a \in EAB$ such that $pg_{a'} = pg_a$, $o_{a'} = o_a$, $m_{a'} \leq m_a$ and $\sim \tau_{a'} \vee \tau_a$ is a tautology. \square

We next show that the use of safe authorizations maintains authorization consistency.

Theorem 1. If all the authorizations on derived objects are derived using the safe authorization derivation rule, then EAB enjoys authorization consistency.

Proof: Assume all authorizations on derived objects are derived using the safe authorization derivation rule. Let us assume that there exists an inverse authorization a' . To prove the theorem, we need to prove that there exists an authorization $a \in EAB$ such that $pg_{a'} = pg_a$, $o_{a'} = o_a$, $m_{a'} \leq m_a$ and $\sim \tau_{a'} \vee \tau_a$ is a tautology.

According to definition 13, for a' to exist, there must be a derived authorization a'' and a rule authorization $ra = (pg_{ra}, fr_a(P), m_{ra}, +, \tau_{ra})$ such that $read \leq_{\mu} m_{a''}$,

$pg_{a'} = pg_a = pg_{ra}$, $m_a \leq_\mu read$, and $o_{a'} \in P$ if f_{ra} is of reconstructive type or $o_{a'} \in \{P - SUBP\}$ if f_{ra} is of partial reconstructive type. Moreover, $\tau_{a''} = (\tau_{ra} \wedge \tau_{a'})$.

According to definitions 11 and 12, $m_{a''} \leq_\mu lub(\cup_{o \in PM_{pg,o}^+})$. That is, $m_{a''}$ is never higher than the least upper bound of the privilege modes of the authorizations relevant to the objects involved in computing o'' . This includes the object o in a . Hence $m_{a'} \leq m_{a''} \leq m_a$.

Also, from these two definitions, we know that $\sim \tau_{a''} \vee (\bigwedge_{a \in rel(o)+} \tau_a)$ is a tautology. Therefore, $\bigwedge_{a \in rel(o)+} (\sim \tau_{a''} \vee \tau_a)$ is a tautology. For that to be true, $\forall a \in rel(o)$, $(\sim \tau_{a''} \vee \tau_a)$ is a tautology, which concludes the proof. \square

6. ACCESS CONTROL

In this section we provide two types of access requests (Section 6.1) to be followed by an algorithm for evaluating access requests (Section 6.2).

6.1 Types of access requests

An access request can be of two types. A user may follow the traditional access control method and request access for a specific object at a certain time point. This we refer to as a *point access request*. Alternatively, a user may request access for an object for a specific duration, which we refer to as an *interval access request*. The latter is useful, for example, in cases where information is provided continuously (as in data streaming) and the service is charged accordingly. Formally:

Definition 15. [Access Request] A point access request is a triple $car = (s, o, m)$. An interval access request is a quadruple $iar = (d, s, o, m)$, where $d \in \mathbf{N}$. \square

Each access request is timestamped with the time for which access is requested by a subject s with privilege mode m on object o (annotated as $t_{req} \in \mathcal{T}$). t_{req} serves as a decision factor whenever relative temporal expressions are involved, e.g., $t_x + 5minutes \leq t_{req}$. An interval access request is interpreted as an access request by a subject s with privilege mode m on object o for duration $[t_{req}, t_{req} + d) \in \bar{\mathcal{T}}$. Therefore, the data that is streamed to the user must be refreshed with relevant changes to the database state and in accordance with the current time. It is worth noting that an access request in our model is a natural extension of an access request in authorization models for non-temporal data. Therefore, a point access request is interpreted the same way as in authorization models for non-temporal data whenever the authorization base does not enforce temporal constraints (effectively meaning that τ is a tautology). Also, an interval access request, where $d = 1$, is equivalent to a point access request. It is also noteworthy that the access granting decision is binary with respect to any given time point, yet it may provide partial access for intervals, i.e., access to a sub-interval only. A final note relates to the case where $d = \infty$. This case is interpreted as “the access request holds until it is explicitly terminated.” This type of authorization is useful whenever stream data is involved, as is the case with streaming stock exchange data to a user over the Web. In this case, the closing of a session (either in an orderly fashion or abruptly) serves as an explicit termination of the authorization.

Given an access request $car = (s, o, m)$ at time t_{req} , we should first verify the *presence of authorizations*, i.e., to determine whether there exists an authorization $\mathbf{a} = (pg, o, m, +, \tau)$, such that $s \in pg$. We next provide a *formula assessment*. This

step involves evaluating the aggregated temporal impact of relevant authorizations (positive and negative) and generating an aggregated formula τ' for which such an authorization can be granted. The next step involves *state-elements selection*, where a state-element se of o is selected only if $S \models_{VA(se.t_x, se.t_s, se.t_e, t_{req}, se.val)} \tau$.

An interval access request $iar = (d, s, o, m)$ at time t_{req} can be conceptually viewed as d separate point access requests, $\{car_i = (s, o, m)\}_{i=0}^d$, timestamped with $\{t_{req}^i = t_{req} + i\}_{i=0}^d$. Each car_i is possibly granted with a different set of state-elements of o , such that τ is satisfied with respect to a t_{req}^i assignment. We shall refer to the process of substituting one set of state-elements with another as *version switching*. It is important to note at this point that access to a state-element may be revoked whenever the temporal characteristics of the state-element no longer satisfy τ . In most cases the actual revocation time can be computed as soon as access is granted. For example, consider an authorization

$$(pg, o, read, +, (t_x + 5minutes \leq t_{req} \wedge t_{req} \leq t_e + 5minutes)). \quad (11)$$

For the state-element se_1 in Example 5 above, a request time $t_{req} = 63$, and a request interval $[t_{req}, t_{req} + 10)$, a request can be granted for the interval $[63, 68)$. The lower bound of this interval (63) is computed by the assignment $t_x + 5minutes = 58 + 5 = 63$ (since $se_1.t_x = 58$). The upper bound of the interval (68) is computed by the assignment $t_e + 5minutes = 63 + 5 = 68$.

6.2 Evaluating access requests

In this section we introduce an algorithm for evaluating access requests. We first present a practical approach towards validating whether given a state-element se ,

$$S \models_{VA(se.t_x, se.t_s, se.t_e, t_{req}, se.val)} \tau.$$

Such an approach takes an advantage of the natural indexing mechanism of temporal databases, where state-elements are stored in order of transaction time, and additional indices may exist on the start and end times of a valid time. Therefore, we establish separate constraints on t_s, t_e, t_r , and t_x (τ_s, τ_e, τ_r and τ_x , respectively), using basic arithmetic manipulations. For example, given an authorization $(pg, o, read, +, t_x + 5minutes \leq t_{req})$, the temporal expression is segmented, a process which results in a single constraint (no constraint is set on t_s or t_e since they are not present in τ):

$$\tau_x = t_x \leq t_{req} - 5$$

In the example given above, τ_x resulted in a constraint of the form $t_o \diamond t$, where $t_o \in \{t_x, t_s, t_e, t_r\}$, \diamond is a comparison operator ($=, \geq, etc.$), and t is a time point. This type of constraint is definitely not the only possible constraint type. As an example, consider an authorization \mathbf{a} such that $\tau_a = t_x \leq t_s + 5$, that is, all the state-elements that become valid only 5 minutes after being committed to the database. Applying segmentation to τ_a , results in two constraints, as follows:

$$\begin{aligned} \tau_x &= (t_x \leq t_s + 5) \\ \tau_s &= (t_s \geq t_x - 5) \end{aligned}$$

Clearly, any state-element that satisfies τ_x also satisfies τ_s . Removing redundant constraints is based on efficiency considerations, *e.g.*, for which temporal dimen-

```

Input:  $(s, o, m)$  or  $(d, s, o, m)$ ,  $t_{req}$ , and  $\{se\} \in o$ 
Output:  $\{(s, se, m, \bar{t})\}$ 
Procedure:
/* presence of authorization */
  If there exists an authorization  $a = (pg, o, m, +, \tau) \in EAB$ 
  such that  $s \in pg$ , then
     $\tau' = null$ 
/* formula assessment */
  If there exists an authorization  $a = (pg, o, m, -, \tau) \in EAB$ 
  such that  $s \in pg$ , then
     $\tau' = \tau' \wedge \neg \tau$ 
/* state-elements selection */
  If  $d = null$  then  $d = 1$ 
  repeat
     $SE = \{se \in o \mid \tau_s \wedge \tau_e \wedge \tau_r \wedge \tau_x\}$ 
/* version switching */
  Foreach  $se_i \in SE$ 
     $\bar{t}_i = bind(se_i.t_s, se_i.t_e, se_i.t_r, se_i.t_x) \cap [t_{req}, t_{req} + d)$ 
    If  $\bar{t}_i \neq \emptyset$  then output  $(s, se_i, m, \bar{t}_i)$ 
  end For
  until  $now \geq t_{req} + d$ 
end If

```

Fig. 4. Access request evaluation

sion the database maintains indices, and whether a specific index is optimized for equality retrieval or a range retrieval.

We have omitted the details of the segmentation process. The process is simple in its logic, yet tedious in its structure, and introducing it does not contribute much to the essence of this paper. It is also worth noting that segmentation is performed on individual authorizations and needs to be performed for each modified authorization, *e.g.*, in case of revocation.

Figure 4 presents the algorithm for evaluating access requests. The algorithm accepts as an input an access request ar for an object o (either a point access request (s, o, m) or an interval access request (d, s, o, m)), its timestamp (t_{req}) and a continuous feed of state-elements of o . The output consists of a set of state-elements, authorized for access, and a time interval during which each state-element is authorized for ar . The algorithm is invoked once per access request and is terminated when one of the following two conditions is fulfilled:

- (1) No authorization $\mathbf{a} = (pg, o, m, +, \tau) \in EAB$ exists.
- (2) The request interval $[t_{req}, t_{req} + d)$ has expired.

The algorithm uses the segmented constraints τ_s, τ_e , and τ_x , as computed above. For example, consider an authorization request $(s, o, read)$, issued at time $t_{req} = 63$ and assume that $s \in pg$. As a first step, assume the authorization $(pg, o, read, +, t_x + 5 \text{minutes} \leq t_{req})$ is retrieved. Given $t_{req} = 63$, one has that

$$\begin{aligned}
 \tau_x &= t_x \leq t_{req} - 5 \\
 &= t_x \leq 63 - 5 \\
 &= t_x \leq 58
 \end{aligned}$$

Once establishing constraints on the temporal dimensions of the state-elements, a set of state-elements that satisfy these constraints is computed. Using the example given above, and the set of state-elements from Example 5, the only state-element that qualifies for the set is se_1 , with $t_x = 58 \leq 58$. se_2 , fails to satisfy the constraint with a transaction time of 64.

Consider now an interval authorization request of the form $(10, s, o, read)$. Again, the authorization $(pg, o, read, +, t_x + 5\text{minutes} \leq t_{req})$ is retrieved. Given an interval $[t_{req} = 63, t_{req} + d = 63 + 10 = 73)$, τ_x is computed as follows:

$$\begin{aligned} \tau_x &= (t_x \leq 58 \text{ (computed by substituting } t_{req}/63)) \\ &\cup t_x \leq 59 \text{ (computed by substituting } t_{req}/63 + 1) \\ &\dots \\ &\cup t_x \leq 68 \text{ (computed by substituting } t_{req}/63 + 10) \\ &= (t_x \leq 68) \end{aligned}$$

In this case, both se_1 and se_2 qualify for retrieval.

The next step involves the version switching. For each state-element in the selected set, an interval during which it can be presented to the user is computed, using $bind(se_i.t_s, se_i.t_e, se_i.t_x)$. Following the example given above, we next compute the access interval for se_1 , by substituting $t_x/58$ in $t_x + 5\text{minutes} \leq t_{req}$, to result in $[63, \infty)$ (since t_{req} only provide a lower bound in this case). For se_2 , the substitution $t_x/64$ results in an interval $[69, \infty)$. These intervals are further truncated by the request interval $[63, 73)$, since $t_{req} = 63$ and $t_{req} + d = 63 + 10 = 73$. Therefore, the interval associated with se_1 is $[63, 73)$, and that of se_2 is $[69, 73)$. It is worth emphasizing that the combination of τ and the interval $[t_{req}, t_{req} + d)$ provides the needed mechanism to forbid access beyond that specified by τ and the access request, even if the validity of the state-element goes beyond it.

7. DESIGN AND IMPLEMENTATION

This section briefly describes several issues relating to the design and implementation of an authorization model for information portals. It covers issues related to the maintenance of the authorization base and the temporal database. We also provide a brief overview of a running demo based on this model.

Authorization base evolution. Once the expanded authorization base has been created, its maintenance can be performed in an incremental fashion. Therefore, for each newly inserted authorization \mathbf{a} , the algorithm in Figure 3 is performed, using \mathbf{a} as an input. Revocation of an authorization requires performing the following steps. Let $(pg, o, m, sign, \tau)$ be the revoked authorization:

- (1) If there exists an authorization $(pg, o, m, sign, \tau') \in EAB$, it should be replaced with an authorization $(pg, o, m, sign, \tau'')$, such that $\tau' = \tau'' \vee \tau$. It is worth noting that we are guaranteed to have τ'' , due to the way the expansion algorithm was defined. Also, from the unification process we can establish that the expanded authorization base EAB cannot contain two identical authorizations $(pg, o, m, sign, \tau)$. For example, consider the following two authorizations:

$$\begin{aligned} \mathbf{a}_1 &= (pg, o, write, +, t_x + 5\text{minutes} \leq t_{req}) \\ \mathbf{a}_2 &= (pg, o, write, +, t_v + 9\text{minutes} \leq t_{req}) \end{aligned}$$

The expansion algorithm would generate the authorization $(pg, o, write, +, (t_x + 5minutes \leq t_{req} \wedge t_{req} \leq t_v + 9minutes))$. In the case \mathbf{a}_1 is removed from the authorization base, $t_x + 5minutes \leq t_{req}$ should be removed from the expanded τ . Therefore, the expanded authorization is reduced to \mathbf{a}_2 , as indeed it should.

- (2) Using DFS, we identify any authorization on the mode/privileged group that may be affected by this change. Therefore, if there exists an authorization $(pg, o, m', sign, \tau') \in EAB$ such that $m' \rightarrow m \in E_\mu$ (in the case of a positive authorization) or $m \rightarrow m' \in E_\mu$ (in the case of a negative authorization), it is modified to be $(pg, o, m', sign, \tau'')$, where $\tau' = \tau'' \vee \tau$, unless there exists an authorization $(pg, o, m', sign, \tau^* \vee \tau) \in AB$ for some τ^* . In this case, τ is explicitly supported by an authorization in AB , and therefore $(pg, o, m', sign, \tau')$ remains unchanged. The same holds for the π lattice.

Another consideration, associated with authorization base evolution, involves active interval access requests. Obviously, we need to be concerned with relevant authorizations only, *i.e.*, authorizations of the type $\mathbf{a} = (pg, o, m, +, \tau) \in EAB$ or of the type $\mathbf{a} = (pg, o, m, -, \tau) \in EAB$. Whenever a relevant authorization is either added or revoked, if $[t_{req}, t_{req} + d)$ has not yet expired, τ' should be re-computed and re-evaluated against the existing set of state-elements. Having done that, the algorithm continues its course for the remaining of the request interval. Clearly, the modification of τ' can be efficiently performed in an incremental way.

Temporal database evolution. We are concerned with the monitoring of new state-elements once access is granted. Unless an access request is a point access request, the duration of the access spans into the future and therefore may include new state-elements as they come along. For example, consider Example 5. At time 176, se_3 was introduced with $se_3.t_x = 176$ and $se_3.t_v = [175, 200)$. For an authorization of the form $(pg, o, read, +, t_x + 5minutes \leq t_{req})$ and an interval authorization request of the form $(150, s, o, read)$, such that $t_{req} = 63$, se_3 becomes eligible for this access request at time 181. Therefore, one may view the algorithm in Figure 4 as a generator of access permissions, which are later being fed to a calendar-based utility that is responsible for the version switching process. For any new state-element, the algorithm identifies the time interval for which an access can be granted (\bar{t}) and feeds the calendar-based utility with the state-elements identification and the start and end times of access granting.

Demo. Figure 5 provides a screen snapshot, in which real-time data⁹ is contrasted with delayed data, as specified by $\tau = t_x + 15minutes \leq t_{req}$. It is worth noting that data accumulates over time since there is no constraint over the valid time dimension. To achieve the effect of a “moving window,” in which only the most recent data is available for viewing, the wff should be modified to be $\tau = (t_x + Xminutes \leq t_{req} \wedge t_{req} - Xminutes \leq t_e)$, where $Xminutes$ serves as the moving window size.

The demo is available at http://business.rutgers.edu:4610/sign_in.html and allows registered users to receive data regarding up to two stocks with various time delays. The demo main screen is partitioned into two parts (see Figure 5), where

⁹The “real-time data” is actually a 20 minutes old data, as provided by Yahoo.

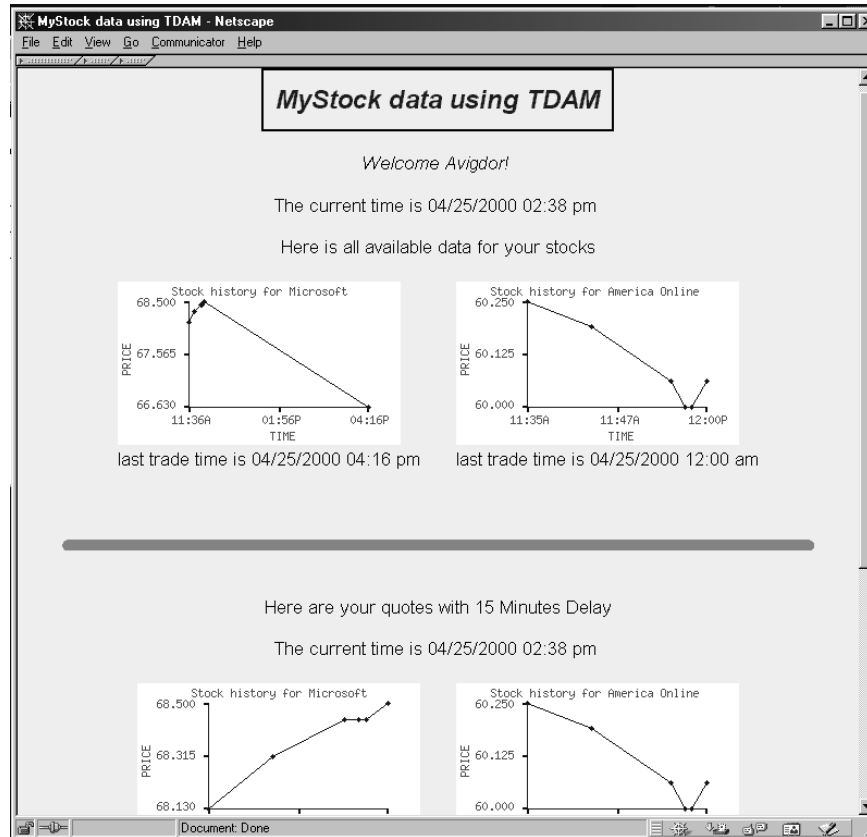


Fig. 5. A snapshot of the authorization model demo

the top provides the most current data while the bottom one provides the delayed data. The data is collected using a backend Java program into a MySQL database. An Apache Web server provides Web-based access to users. Finally, a CGI script is utilized in accessing the database and requesting the appropriate stream of data.

Another demo, available at <http://128.6.67.225/EWSNews/html/cs541.html>, allows users to add new authorizations and test them against a simple database. Both systems demonstrate the capabilities of temporal authorizations, and do not address the issue of derived authorizations.

8. RELATED WORK

In the basic authorization model, the authorization is specified using a triple $\langle s, o, p \rangle$, interpreted as “subject s is authorized to exercise privilege p on object o .” Recently, several extensions to this basic authorization model have been suggested. One extension deals with increasing the expressive power of the authorization models and developing appropriate tools and mechanisms to support them. This extension includes negative authorization [Bertino et al. 1993a; Rabitti et al. 1991], role-based authorizations, task-based authorizations, separation of duties [Clark and

Wilson 1987; Sandhu 1988; Sandhu 1991], and temporal authorization [Thomas and Sandhu 1993; Bertino et al. 1994; Bertino et al. 1996]. Another extension deals with authorization models for advanced data management systems such as object-oriented DBMSs [Fernandez et al. 1989; Spooner 1989; Rabitti et al. 1991; Bertino et al. 1993b], distributed DBMSs [Woo and Lam 1992; Woo and Lam 1993; Samarati et al. 1994], WfMSs (workflow management systems) [Atluri and Huang 1996; Atluri and Huang 1997; Bertino et al. 1997], and hypertext systems [Samarati et al. 1996].

In this paper, we extend the basic authorization model in two directions – (i) the capability to express authorizations based on the temporal attributes associated with data, such as transaction time and valid time, and (ii) the derivation of authorizations for derived data based on the authorizations of the base data from which the data is derived. We next discuss the related research in these two areas.

Of the aforementioned models, the only authorization model that allows specification of temporal parameters, is the *temporal authorization model* proposed by Bertino et al. [Bertino et al. 1996]. In this model, an authorization is specified as $(\mathbf{time}, \mathbf{auth})$, where $\mathbf{time} = [t_b, t_e]$ is a time interval, and $\mathbf{auth} = (\mathbf{s}, \mathbf{o}, \mathbf{m}, \mathbf{pn}, \mathbf{g})$ is an authorization. Here, t_b and t_e represent the start and end times during which \mathbf{auth} is valid, respectively. \mathbf{s} represents the subject, \mathbf{o} the object, and \mathbf{m} the privilege. \mathbf{pn} is a binary parameter indicating whether an authorization is negative or positive, and \mathbf{g} represents the grantor of the authorization. This model also allows such operations as **WHENEVER**, **ASLONGAS**, **WHENEVERNOT**, and **UNLESS** on authorizations, where, for example, **WHENEVER** can be used to express that a subject s_i can gain privilege on object o **WHENEVER** another subject s_j has the same privilege on o . Later, Bertino et al. [Bertino et al. 1998] have extended the temporal authorization model to support periodic authorization. The extended model allows specification of authorizations with a predefined time interval but does not derive them based on the temporal attributes of the data. Therefore, it is not sufficiently expressive to serve as an access control model for the types of data in the information portals. As an example, the model as proposed in this paper, can express security policies such as “a subject s is allowed to read object o one month after it has been written,” and “only managers are allowed to read object o which is valid during a future time interval.”

With respect to addressing the authorization models for derived data, relevant research includes the work in the area of federated databases, data warehouses and views; the research that is most closely related to our work is that by Rosenthal et al. [Rosenthal and Sciore 1998; Rosenthal et al. 1999]. Our work is consistent with the approach taken in Rosenthal et al. [Rosenthal and Sciore 1998], in which authorizations of base data are being propagated to views. However, no specific mechanisms are given to support propagation based on either temporal aspects of data or the accessibility of subjects to view derivation rules. In [Rosenthal et al. 1999], ideas regarding the derivation of authorizations of derived data were given. We provide a formal model that defines the notion of authorization consistency expressed in terms of the authorizations of the base data. Moreover, authorizations in our model are based on the temporal attributes of data. Furthermore, we remove an assumption that view definitions are world-readable. Finally, in [Rosenthal and Sciore 1999], the authors mention function invertability (in parallel with our notion

of reconstructive mappings) as a possible future research. We extend the security aspect of such functions to include hybrid reconstructive functions as well. In [Wang and Spooner 1987], an authorization model that is based on views and an ownership paradigm is supported. The model supports read-only access to views and is thus far more limited than the model we have introduced herein.

Several models were suggested for the support of authorization needs of federated databases, including [Templeton et al. 1987], [Jonscher and Dittrich 1994], [Blaustein et al. 1995], and [di Vimercati and Samarati 1997], that are relevant to our work; however, none of them support authorization based on temporal characteristics of data. Mermaid [Templeton et al. 1987] is a front-end system to integrate multiple homogeneous relational DBMSs. It supports an authorization model that ensures the autonomy of databases authorization schemes and there is no attempt to relate various properties through replication and derivation mechanisms, to ensure authorization consistency. The model suggested in [Jonscher and Dittrich 1994], while supporting consistent authorizations of replica, does not support materialized views per-se. We elaborate on the matter of access rights to computation methods, not just as far as execution is concerned (as was suggested in [Jonscher and Dittrich 1994]) but also in terms of rewriting methods. The approach in this paper is closely related to the model in [di Vimercati and Samarati 1997]. Both works identify the need to analyze data replica and materialized views (referred to as imported objects and composite objects, respectively, in [di Vimercati and Samarati 1997]) when granting authorizations in a federation, although no explicit reference to mapping authorizations is given in [di Vimercati and Samarati 1997]. Also, our work is an extension to this work in that it provides an algorithm for authorization derivation, while supporting the notions of global and local authorizations. Finally, [Blaustein et al. 1995] presents the system aspects of secure database federations while our work focuses on the modeling of authorizations.

9. CONCLUSIONS

In this paper we have described an authorization model for information portals, where authorizations are derived based on the temporal characteristics of data as well as its relationships with other data, as reflected in derivation mappings. While existing authorization models provide adequate support for conventional databases, we are of the opinion that no authorization model can be found in the literature that is capable of providing access control to data that is both temporal and derived. Hence, the contribution of the paper is both in presenting an authorization model suitable of providing access control to information portals.

ACKNOWLEDGMENTS

We would like to thank George Bertele, Digant Modha, Gang Xu, Jixin Li, Xinyun Yu, and Hao Sun for implementing the demos.

REFERENCES

- ATLURI, V. AND HUANG, W.-K. 1996. An Authorization Model for Workflows. In *Proceedings of the Fifth European Symposium on Research in Computer Security, in Lecture Notes in Computer Science, No.1146, Springer-Verlag* (September 1996), pp. 44–64.

- ATLURI, V. AND HUANG, W.-K. 1997. Enforcing Mandatory and Discretionary Security in Workflow Management Systems. *Journal of Computer Security* 5, 4, 303–339.
- BERTINO, E., BETTINI, C., FERRARI, E., AND SAMARATI, P. 1996. A temporal access control mechanism for database systems. *IEEE Transactions on Knowledge and Data Engineering* 8, 1, 67–80.
- BERTINO, E., BETTINI, C., FERRARI, E., AND SAMARATI, P. 1998. An access control model supporting periodicity constraints and temporal reasoning. *ACM Transactions on Database Systems (TODS)* 23, 3, 231–285.
- BERTINO, E., BETTINI, C., AND SAMARATI, P. 1994. A temporal authorization model. In *Proc. Second ACM Conference on Computer and Communications Security* (Fairfax, VA, November 1994), pp. 126–135.
- BERTINO, E., FERRARI, E., AND ATLURI, V. 1997. A Flexible Model Supporting the Specification and Enforcement of Role-based Authorizations in Workflow Management Systems. In *Proc. second ACM Workshop on Role-based Access Control* (November 1997), pp. 1 – 12.
- BERTINO, E., SAMARATI, P., AND JAJODIA, S. 1993a. Authorizations in relational database management systems. In *Proc. First ACM Conference on Computer and Communications Security* (Fairfax, VA, November 1993), pp. 130 – 139.
- BERTINO, E., SAMARATI, P., AND JAJODIA, S. 1993b. High assurance discretionary access control for object bases. In *Proc. First ACM Conference on Computer and Communications Security* (Fairfax, VA, November 1993), pp. 140 – 150.
- BLAUSTEIN, B., MCCOLLUM, C., NOTARGIACOMO, L., SMITH, K., AND GRAUBART, R. 1995. Autonomy and confidentiality: Secure federated data management. In *Proceedings of the Second International Workshop on Next Generation Information Technologies and Systems (NGITS '95)* (Nahariya, Israel, June 1995), pp. 59–68.
- CLARK, D. D. AND WILSON, D. R. 1987. A comparison of commercial and military computer security policies. In *Proc. IEEE Symposium on Security and Privacy* (Oakland, California, April 1987), pp. 184–194.
- CLIFFORD, J. AND TANSEL, A. U. 1985. On an algebra for historical relational databases: Two views. In *Proc. ACM SIGMOD* (May 1985), pp. 247–265.
- DI VIMERCATI, S. D. C. AND SAMARATI, P. 1997. Authorization specification and enforcement in federated database systems. *Journal of Computer Security* 5, 2, 155–188.
- ETZION, O. 1993. PARDES — a data-driven oriented active database model. *ACM SIGMOD Record* 22, 1 (Mar), 7–14.
- FERNANDEZ, E. B., GUEDES, E., AND SONG, H. 1989. A Security Model for Object-oriented Databases. In *Proc. IEEE Symposium on Security and Privacy* (May 1989), pp. 110–115.
- GAL, A. 1999. Semantic interoperability in information services: Experiencing with COOPWARE. *ACM SIGMOD Record* 28, 1, 68–75.
- GAL, A. AND ATLURI, V. 2000. An authorization model for temporal data. In *Proc. 7th ACM Conference on Computer and Communication Security* (Athens, Greece, Nov. 2000), pp. 144–153.
- GAL, A., ETZION, O., AND SEGEV, A. 1996. TALE — a Temporal Active Language and Execution model. In P. CONSTANTOPOULOS, J. MYLOPOULOS, AND Y. VASSILIOU Eds., *Advanced Information Systems Engineering*, pp. 60–81. Springer.
- JENSEN, C., CLIFFORD, J., GADIA, S., SEGEV, A., AND SNODGRASS, R. 1992. A glossary of temporal database concepts. *ACM SIGMOD Record* 21, 3, 35–43.
- JONSCHER, D. AND DITTRICH, K. 1994. An approach for building secure database federations. In J. BOCCA, M. JARKE, AND C. ZANIOLO Eds., *Proceedings of 20th International Conference on Very Large Data Bases (VLDB'94)* (Santiago de Chile, Chile, 1994), pp. 24–35. Morgan Kaufmann.
- PISSINO, N., SNODGRASS, R., ELMASRI, R., MUMICK, I., OZSU, M., PERNICI, B., SEGEV, A., AND THEODOULIDIS, B. 1994. Towards an infrastructure for temporal databases—A workshop report. *ACM SIGMOD Record* 23, 1, 35.

- RABITTI, F., BERTINO, E., KIM, W., AND WOELK, D. 1991. A model of authorization for next-generation database systems. *ACM Transactions on Database Systems* 16, 1 (March), 88–131.
- ROSENTHAL, A. AND SCIORE, E. 1998. Propagating integrity information among interrelated databases. In *Proc. IFIP 11.6 Workshop on Data Integrity and Control* (Warrenton VA., 1998), pp. 5 – 18.
- ROSENTHAL, A. AND SCIORE, E. 1999. First class views: A key to user-centered computing. *SIGMOD Record* 28, 3, 22–28.
- ROSENTHAL, A., SCIORE, E., AND DOSHI, V. 1999. Security administration for federations, warehouses, and other derived data. In *Research Advances in Database and Information Systems Security* (1999), pp. 209–223.
- SAMARATI, P., AMMANN, P., AND JAJODIA, S. 1994. Propagation of authorizations in distributed database systems. In *Proc. Second ACM Conference on Computer and Communications Security* (Fairfax, VA, November 1994), pp. 136 – 147.
- SAMARATI, P., BERTINO, E., AND JAJODIA, S. 1996. An Authorization Model for a Distributed Hypertext System. *IEEE Transactions on Knowledge and Data Engineering* 8, 4, 555–562.
- SANDHU, R. S. 1988. Transaction Control Expressions for Separation of Duties. In *Proc. Fourth Computer Security Applications Conference* (1988), pp. 282–286.
- SANDHU, R. S. 1991. Separation of Duties in Computerized Information Systems. In S. JAJODIA AND C. LANDWEHR Eds., *Database Security, IV: Status and Prospects* (1991), pp. 179–189. North Holland.
- SANDHU, R. S. ET AL. 1996. Role-based Access Control Models. *IEEE Computer* 29, 2 (February), 38–47.
- SHOHAM, Y. 1988. *Reasoning about change: Time and Causation from the standpoint of Artificial Intelligence*. MIT press.
- SPOONER, D. L. 1989. The Impact of Inheritance on Security in Object-oriented Database Systems. In *Database Security, II: Status and Prospects*, Carl E. Landwehr, ed., North-Holland, Amsterdam (1989), pp. 141–160.
- TEMPLETON, M., LUND, E., AND WARD, P. 1987. Pragmatics of access control in Mermaid. *Data Engineering* 10, 3 (Sept.), 33–38. Special Issue on Federated database Systems.
- THOMAS, R. K. AND SANDHU, R. S. 1993. Discretionary Access Control in Object-oriented Databases. In *Proc. 16th National Computer Security Conference* (Baltimore, MD, September 1993), pp. 63–74.
- WANG, C. AND SPOONER, D. 1987. Access control in a heterogeneous distributed database management system. In *Proc. IEEE 6th Symposium on Reliability in Distributed Software and Database Systems* (Williamsburg, VA, March 1987), pp. 84–92.
- WOO, T. Y. AND LAM, S. S. 1992. Authorization in Distributed Systems: A Formal Approach. In *Proc. IEEE Symposium on Security and Privacy* (Oakland, California, May 1992), pp. 33–50.
- WOO, T. Y. AND LAM, S. S. 1993. A framework for distributed authorization. In *Proc. First ACM Conference on Computer and Communications Security* (Fairfax, VA, November 1993), pp. 112 – 118.