Authorization Specification and Enforcement in Federated Database Systems*

Sabrina De Capitani di Vimercati   Pierangela Samarati
Dipartimento di Scienze dell'Informazione
Università di Milano
via Comelico 39/41 Milano 20135, Italy
Phone: +39-2-55006257
Fax: +39-2-55006253
{decapita,samarati}@dsi.unimi.it

Abstract

A federated system is a collection of cooperating autonomous databases. Federated systems represent today one of the new emerging technology for data management. This success comes from the need to integrate and work on different existing systems independently developed and evolved. The necessity of making them available to users as if they were a single system, while at the same time not affecting their independent working, arises several issues with respect to authorization management and specification and to access control enforcement. In this paper we discuss some of these issues and present an authorization model for the specification and enforcement of authorizations in federated database systems. The model allows users to make their data available to the federation and to choose among different administrative policies for regulating the specification of authorizations.

Keywords: Federated systems, access control, authorization administration, authorization autonomy.

1 Introduction

Traditional distributed database systems were designed and organized with distribution in mind [6]. When a system had to be developed the global database schema was defined and split into different pieces which were then stored at different, geographically distributed, sites. Today most distributed systems do not fall in this framework. Very often organizations are faced with the need to integrate and cooperate in accessing different databases which have been independently developed and evolved, in order to allow users to access them as if they were a single system. The traditional distributed database

*Preliminary versions of this paper appeared in the *Proceedings of the Fourth European Symposium on Research in Computer Security (ESORICS'96)*, Rome, Italy, September 1996 [7] and in the *Proceedings of the ACM SIGSAC New Security Paradigms Workshop*, Lake Arrowhead, CA, August 1996 [8].

design approach, which would require to redesign from scratch the whole schema, is in this case not applicable. Moreover, the integration process must not affect the different databases, which may need to continue working independently to satisfy the requests of their local users. To solve this a new distributed architecture has been introduced, that of *federated systems*.

A federated system integrates existing, possibly heterogeneous, databases while preserving their autonomy [25]. The main difference between federated systems and traditional distributed systems is that in federated systems each component remains autonomous. Autonomy of a component system means that the local administrator maintains some control over his/her system. Different types of autonomy have been distinguished: *design autonomy*, *communication autonomy*, *execution autonomy*, and *association autonomy*. Design autonomy refers to the ability of a component system to choose how the information it stores is to be organized and accessed, for example, with reference to the data model and query language to be used. Communication autonomy refers to the ability of a component system to decide whether to communicate with (i.e., to respond to the requests of) other component systems. Execution autonomy refers to the ability of a component system to execute local operations without interference from external operations. Association autonomy refers to the ability of a component system to decide whether and how to share its functionalities (operations and resources) with other systems.

In this paper we focus on two particular subtypes of association autonomy: *authentication* and *authorization* autonomy. By authentication autonomy we mean the ability of a system to decide how the identity of each user accessing data stored at the system is established. By authorization autonomy we mean the ability of a system to specify which accesses are to be allowed or denied on objects stored at the system.[1]

Authorization and access control issues have not been receiving, if not for few exceptions, much attention by the research, mainly focused on the various problems related to data access and communication [25]. Although recent research has addressed the problem of protecting federated systems [3, 13, 14, 18, 19, 28, 29, 30] and few federated systems, like Mermaid [26], Orion-2 [16], or the one proposed by Heimbigner and McLeod [10] support some form of authorization specification and access control, several issues still remain to be investigated. However, many are the security issues that need investigation. The need to share data in the federation on one side and to maintain system autonomy on the other side raise several protection requirements which traditional authorization models [5, 23] do not address. Security enforcement at the global level must take into consideration the protection requirements and policies of each participating system. This task can be further complicated by the heterogeneity of the component systems, which may enforce protection policies either difficult to combine or inconsistent with each other. Moreover, local autonomy impacts the ability of the federation to acquire and replicate data or to make them available to others. A major problem in this context is also the establishment of administrative policies that determine the

[1] The need to explicitly consider security issues in a separate category has been also pointed out in [14] where the notion of authorization autonomy is introduced. Note, however, that the notion introduced in [14] corresponds to our notion of authentication autonomy.

authority of the different federation participants for the specification of access authorizations. As a matter of fact, while in a centralized or distributed system ownership or centralized administration may be satisfactory solutions, federated systems call for more flexible approaches [22]. Enforcing strict ownership would put on the data owner the burden of specifying authorizations for federated users and therefore to maintain information on who can access the federation. Applying a centralized administration approach at the global level may imply a loss of control, and therefore of autonomy, for the data owner. Moreover, even traditional problems, such as authentication, require careful reconsideration in the federated context.

In this paper we discuss some of the protection issues which arise in federated systems and discuss possible approaches to their solution. We then present an authorization model for the protection of information in federated systems based on a tightly coupled architecture. The reason for considering a tightly coupled federation is to avoid requiring the users to maintain the relationships with each single system participating in the federation. A tightly coupled architecture allows to rely on a central authority for defining and maintaining the federation. However, even with this centralized administration we allow a good degree of autonomy to each single system participating in the federation. In particular, our model allows users to selectively share their objects by specifying for each objects the access modes that are available for the federation. Different kinds of administrative policies are also supported whereby users can either: *i)* retain the privilege of specifying authorizations on objects, *ii)* delegate the federation administrator the task of specifying authorization, or *iii)* require that accesses be authorized by both the federation administrator and them in order to be allowed. Access authorizations are specified at both the global level (on federated objects) and the local level (on exported objects). Local authorizations can also be negative. This allows users to retain control on their objects even in the case where access administration has been delegated to the federation administrator. Although in the paper we do not deal with authentication issues, the model assumes that two different authentication options are possible at a system whereby a system can require to authenticate users accessing its data (even if the request comes through the federation) or can accept the identity of the users as communicated by the federation.

The remainder of this paper is organized as follows. Section 2 illustrates some security issues that must be considered in ensuring protection to federated systems. Section 3 presents the reference architecture of the federation for our authorization model and introduces some notations. Section 4 illustrates authorizations and operations for constructing the federated data. Section 5 describes how authorizations are specified at the federation and the local level. Section 6 illustrates access controls enforced, globally and locally, upon submissions of access requests to the federation. Section 7 discuss previous work and compares it with our approach. Finally, Section 8 presents the conclusions and outlines future work.

Security aspect		**Possible approaches**
User's Authentication	federation	• require users to explicitly identify themselves and authenticate them • do not enforce any user's authentication procedure
	local components	• require users to explicitly identify themselves (local authentication) • do not enforce any authentication (global authentication)
Access control	at the federation	• based on the remote system's identity • based on the remote identity of the user • based on the federated identity of the user
	at each local component	• based on the remote system's identity • based on the federation's identity • based on the remote identity of the user • based on the federated identity of the user • based on the local identity of the user
Population of the federation		• direct creation (global objects) • import of objects (imported objects): - by the federation administrator - by the administrators of the objects - through negotiation of the federation administrator and the administrators of the objects
Administration of authorizations for imported objects		• federation administrator • local administrator • cooperative
Specification of authorizations		• global and local authorizations are independent • global authorizations are derived from local ones (`bottom-up derivation`) • local authorizations from global ones are derived (`top-down derivation`)
Global and local authorization consistency		• not required • changes are propagated as soon as they happen (`immediate`) • changes are discovered and propagated during access control (`access time`) • changes are propagated at specific time instants (`periodic`) • changes are propagated upon explicit request of the federation or the local administrator (`explicit request`)

Table 1: Security problems and possible solutions

2 Protection issues in federated systems

In this section we discuss some of the protection issues which arise in federated systems together with possible approaches to their solution. In the discussion, summarized in Table 1, we focus on the problems related to the authorization management and enforcement. Other issues, such as network security, inference and aggregation, or accountability are outside the scope of this paper.

2.1 Authentication

Authentication is the process of certifying the identity of users, or more generally subjects, requiring access to the data. A good user's authentication is the foundation of any access control system [23]. A user's identity determines the accesses that the user will be allowed or denied on the data.

In federated systems multiple parties may be involved in each access and may therefore require, or provide, user's authentication. We can identify at least three different systems involved in an access required by a given user: 1) the user's local system, i.e., the system from which the user connected to the federation; 2) the system of the federation, if federation is based on the tightly couple architecture; and 3) the system/s where the data to which access is required are actually stored.

The question therefore arises of whether a user should identify him/herself, and hence be authenticated, at each of these systems or, instead, some systems can rely on the authentication performed by other systems in the federation.

A first decision to be taken concerns whether users should connect and authenticate themselves to the federation in order to access federated data. If the answer is yes, in order to access the federated objects, a user should own an account at the "federation site". If the answer is no, this account is not necessary and the federation would be accessible by users without any identification and authentication procedure. Note that the latter approach does not imply that no access control will be executed. However, it implies that access control, if any, will need to refer to the identity of the users as authenticated at the system from which they connected to the federation. This may represent a drawback of this solution, as we will see in the next subsection.

A second decision to be taken concerns whether users need to authenticate themselves at each system involved in the access, i.e., where the data to which access is required are actually stored. Again two are the possible options, which we call *local* and *global* authentication respectively. In the local authentication [26] users are required to re-authenticate themselves at each local system. This approach has the advantage that local access decisions of a local system are taken with respect to local identifiers and therefore each system is not required to be informed about remote or federation identities of users. However, it has the disadvantage of requiring multiple identification steps to the users. Users will have in fact to type in login and password for each system involved in the access. This aspect may also compromise the transparency of the system according to which users should exercise access to a federated object without worrying about the specific local objects from which this has been obtained or about their physical location.

In the global authentication [15], users are not required to authenticate themselves at each local system. Their identity (and/or other information needed for access control) is passed to the system by the federation together with the request. The authenticated identity passed to each system can be either the identity of the user at the local system from which he connected or the identity of the user at the federation. The identity to be passed depends also on the authentication option applied at the federated system. Global authentication has the advantage that users will not need to identify themselves at each system. The drawback is that access control at each local system will need to refer to remote or federation identities of users.

2.2 Access control

Access control is the process of checking each user's request against the specified authorizations and granting or denying it accordingly. Major issues to be tackled concern where and with respect to which identity authorization check should be performed.

As for the *where*, we note that access control can be enforced either: at the federation, at each local system, or at both the federation and the local system. Access control at each local system permits to maintain autonomy of each participating system in determining accesses to be allowed or denied on the data it stores. Access control at the global level permits to control requests before splitting them in local requests to be forwarded to the different local systems. This centralized control may avoid unnecessary sending of requests if the federation knows that they will certainly be denied. It also allows to support cases where administration of access authorizations is delegated to the federation administrator.

As for the identity against which access control should be enforced, the solution obviously depends on the authentication policy applied. In particular, access control at the local level can be enforced by considering: *i)* the remote system from which the user connected, *ii)* the identity of the federation, *iii)* the identity of the user at the remote system, *iv)* the federated identity of the user, or *v)* the local identity of the user. In the first two cases access decisions are taken only with respect to the identities of the system or of the federation involved without considering the identity of the user. For instance, at the local system, an authorization can specify that object `o1` can be accessed by any user through federation `f1`. In cases *iii)* and *iv)* identity-based access decisions are taken by the local system but with respect to the user's identity communicated to it by the federation. For instance, an authorization at a given system $\mathtt{s}_3$ can specify that user identified as `bob` at system $\mathtt{s}_2$ can access a given object. This approach, however, has the drawback that authorizations at the local system need to be specified with respect to identities not administered by the local system itself. Finally in case *v)* identity-based decisions are taken by the local system but with reference to user identifiers locally administered. Note that this last solution assumes local authentication.

Approaches to enforce access control at the global level are analogous. In particular, access control can be enforced with reference to: *i)* the system from which the user connected, *ii)* the user's identity at the remote system, or *iii)* the user's identity at the federation. The pros and cons of the different approaches are similar to those discussed above.

2.3 Population of the federation

Populating the federation means defining the objects that are part of the federated schema. How the schema is actually defined and maintained is outside the scope of the access control system. However, the access control system has to regulate the execution of the operations for defining the federated schema to ensure that only authorized subjects will be allowed for them.

Population of the federation can be done in two ways: by directly creating objects in the federated database, or by importing objects from the local systems taking part in the federation.

Direct creation of objects in the federated database can be executed by the federation administrator or by any user explicitly authorized for that.

Import of objects from local systems is more complex, since it requires agreement between the local administrator/s of the objects and the federation administrator: the local administrator/s of an object must be willing to share the object with the federation; the federation administrator must be willing to include the object among the federated data. This negotiation process can be required only once, at the time a system enters a federation, or every time an object is imported. In the first case, there will be only one negotiation phase after which users will be allowed to directly insert their objects in the federation and the federation administrator will be allowed to directly import objects. In the second case, the negotiation process can be carried out through different steps as follows. First, local users declare the objects they wish to share with the federation, thus defining the *export* schema from which the federation administrator can get data. This operation allows simply to declare data which are available to the federation but it does not include them in the federated schema. Second, the federation administrator imports objects into the federation by getting them from the export schemas. This approach has the advantage that negotiation can be enforced at the granularity of each single object. It therefore allows users to selectively share their objects and federation administrators to selectively import objects in the federation. The fact that both the object's and the federation's administrator must agree in order for an object to be inserted in the federation also represents a guarantee to both of them with respect to the protection of the information they manage.

2.4 Administration of authorizations

A major issue that arises after the federation has been populated is who should administer access on the federated objects, i.e., who should specify authorizations to exercise privileges on them. As for objects directly created in the federation, classical administrative policies applied in centralized systems can be considered. For instance, the administration can rest with the federation administrator (*centralized* administration) or with the user who created the object (*ownership*).

Administration of objects imported from local systems is instead more complex.

Should it be left to the federation administrator or to the administrator/s of the local objects imported? It is desirable that a balance be maintained between the necessity of avoiding the local users the burden of specifying authorizations on federated data and the necessity of leaving to the local users some form of control over their objects. Three different approaches can be taken with respect to administration of federated objects. A first approach consists in delegating the administration of the object to the federation administrator. The federation administrator specifies authorizations to access the federated objects and the access control decisions are taken only with respect to these authorizations, i.e., no access control is enforced at the local level. A second approach consists in leaving the privilege of specifying authorizations to the administrator/s of the local object, i.e., the user/s authorized to administer the object locally. In this case, no authorizations need to be specified by the federation administrator and access control decisions are taken only with respect to authorizations specified by the local administrator/s. A third approach consists in allowing both the federation administrator and the local object's administrator/s to specify authorizations. The two administrators therefore cooperate in specifying accesses to be permitted. The different approaches obviously provide different degrees of control of the local user on his data, which imply different degrees of administrative burden on the user. We believe that the indiscriminate application of any of the approaches may result too rigid. Indeed, different situations can be found where any of the approaches can be preferred over the other ones. It would therefore be desirable that the authorization mechanism be able to enforce the different options. The choice of the specific administrative policy to be applied with respect to an object can be the result of a negotiation between the administrator of the local object/system and that of the federation involved.

2.5 Authorizations specification

As already discussed in the previous subsection, authorizations may need to be specified at two different levels: at the global level (on the federated data) and at the local level (on the objects exported to the federation). Different approaches can be taken for the specification and coexistence of global and local authorizations.

The first approach consists in considering the two sets of authorizations as *independent*. The federation administrator specifies global authorizations to access the federated data. The local administrator specifies authorizations to access the local objects. Global and local authorizations are specified independently. However, it is desirable that the specifications be coordinated and that the two administrators cooperate in order to avoid inconsistent specifications.

The other two approaches are based on the assumption that global and local authorizations are related and therefore local authorizations can be derived from the global ones or vice versa. The two approaches differ in the direction of the derivation: *top-down* vs *bottom-up*. With top-down derivation, access authorizations are specified at the global level and then derived at the local level [14]. First, the federation administrator specifies authorizations for users to access global objects at the global level. Then, the accesses to the local data needed for successful execution of the accesses globally authorized are determined. Hence, a request to grant the authorization for each of these accesses is sent to the corresponding local system. At the local system the administrator of the interested object can decide whether to accept or reject the request, i.e., whether to specify the required authorization. If consistency between the global and local authorizations is required, the global authorization will be granted only if all local grant requests are accepted. Consistency means that a request permitted according to the global authorizations cannot fail due to access rejection at the local level. With bottom-up derivation, authorizations at the global level are derived from authorizations at the local level [4]. In this approach, when an object is imported in the federation, global authorizations are derived from the authorizations locally specified by the object's administrator/s. If the federated object is composite, i.e., obtained from more local objects, the authorizations on all the local objects must be considered to derive the global authorizations. The case may arise in which according to the authorizations at some system a given access should be granted, while according to the authorizations at another system the same access should be denied. In this case, if consistency is required, no global authorization will be derived on the federated object. The global authorizations will then mirror the intersection of the privileges allowed by the local authorizations.

A main drawback of the approaches based on derivation is that it might be very difficult, if not impossible, to derive the local authorizations corresponding to a global authorization or vice versa. For example, consider top-down derivation. We know that each global request can be mapped by the data management system onto corresponding local requests on local data. Therefore, local privileges necessary on local objects in order to satisfy a global request are known. The problem is with subjects. As we have already discussed in Section 2.1 subjects of authorizations at the global and at the local level can differ. Global authorizations use identifiers established by the federation administrator. By contrast, in case of local authentication, local systems use local identifiers. Moreover, at both the local as well as the global level user groups or roles can be supported. The knowledge about their configuration (i.e., which users are part of which groups or can assume given roles) is limited to the system/federation where they have been defined and is not known at other systems. As a consequence, authorizations specified at the global level for groups/roles cannot be mapped onto authorizations at the local level for the same groups/roles, since these do not have any meaning outside the federated system. Therefore the problem arises of maintaining the correspondence between global and local subject identifiers. An authorization for a group/role specified at the global level will most probably need to be mapped onto several authorizations at the local systems, one for each user in the group or allowed for the role. Even if this mapping can be enforced (and it may not always be so) the problem arises of maintaining the correspondence between authorizations upon changes in the configuration of groups/roles or in the authorizations (if new authorizations are granted or existing authorizations revoked).

2.6 Consistency between global and local authorizations

If derivation of authorizations, either bottom-up or top-down, is enforced, the problem of maintaining the correspondence between the authorizations upon changes arise. Different propagation strategies, similar to those used for update propagation in distributed databases may be applied for this. In particular, changes to authorizations can be propagated *immediately*, at the *access time*, *periodically*, or upon *explicit request*. In the first case, whenever a change occurs at the global level the corresponding changes necessary at the local level are immediately enforced (the direction is inverse in case of bottom-up derivation). This approach has the advantage of always providing full consistency between the authorizations. However, it has the disadvantage of requiring synchronous changes to authorizations stored at different systems, which may not always be possible (for instance because of system or communication failures). The other approaches overcome this drawback by allowing temporary inconsistencies between the authorizations. In the second approach these inconsistencies will be found out at the access time and only then they will be fixed. For instance, suppose a global authorization is derived from several local authorizations and that after some time one of these authorizations is revoked. Suppose now a user requires to access a federated object and that the access is authorized by the global authorization derived. When the corresponding requests on local objects are sent to the systems the negative response from one of them will inform the federation that the global authorization is no longer supported by local object authorizations and it should therefore be removed. In the third approach changes to authorizations are propagated periodically, for instance every day or every week. Finally, in the latter approach changes to authorizations are propagated upon explicit request by either the federation administrator or the local administrator.

3 The reference system architecture and basic concepts of our model

In this section we illustrate the reference architecture of the federated system and give the basic assumptions on which we base our authorization model.

We consider a tightly coupled federation structured on two levels: at the global, or federation, level there is the federation and at the local level the different systems participating in it. At the global level a privileged user, called *federation administrator* is responsible for creating and maintaining the federated schema and specifying authorizations for users to access the objects in the federation.

At the local level there are the different systems taking part in the federation. We distinguish three categories of participants:

- *customers*. These are systems whose users can be authorized to connect to the federation and access its objects.
- *providers*. These are systems that take part in the construction of the federated data, i.e., which can export their local objects for the population of the federation.
- *customers-providers*. These are systems that fall in both categories above.

At each system, a privileged user, called *system administrator*, is responsible for the relationship with the federations in which the system participates.[2] Registration of a system in a federation is the result of a negotiation between the administrator of the system and the administrator of the federation. In the following, given a federation f notation *providers(f)* and *customers(f)* indicate the systems registered as providers and customers of f, respectively. Systems in the third category above belong to both sets.

In this paper we do not deal with the problem of authenticating the users. However, in the model we consider the different authentication options that can be supported. We assume that each system registered as a provider of a federation can, during the negotiation phase, choose between two different authentication policies according to which the identity of the users accessing local objects through the federation can be established: *global* and *local* authentication (see Section 2.1). Remember that in the *global* authentication federation's users do not need to identify themselves at the local system. By contrast, in the *local* authentication, before processing each access request coming by a user through the federation, the participating system will require the user to identify him/herself.

3.1 Subjects and objects

At each system a set of *local users* is assumed and a set of *local objects* is stored. We do not make any assumption on the data model used at the federation or at each specific system. Given a system s, we refer to the objects stored in it as O_s. Moreover, given an object $o \in O_s$, we denote with M_o the set of access modes executable on it. The specific data types and access modes applicable on the objects will depend on the data model considered. For instance, in a relational model, M_o will be composed of the elementary select, insert, and update operations plus all applications defined for a relation. In an object-oriented model, M_o will be composed of all methods, elementary and non, executable on object o. To make our model independent of the specific administrative policy applied at the local systems, we assume each object o is associated with a set of administrators. The set of administrators associated with an object o contains: the *object's owner*, if ownership is applied; the *system's administrator*, if a centralized policy is applied; all *users owning an administrative authorization* on the object, if decentralized administration is applied.

At the federated system two kinds of subjects are considered: *users* and *groups*. Users are entities allowed to connect to the federation and submit requests to access federated data. Groups, which are sets of users, can be defined with reference to the users's identities at the federation or at the systems from which they can connect to the federation (for instance a group can be defined as containing federated user `tom` plus all users connected to the federation from systems `s1` and `s2`). Each user can belong to zero, one or more groups.

At the global level, three kinds of objects are supported:

[2]For the sake of simplicity we consider a single system administrator to be responsible for the relationship with all the federations in which a system can participate. The approach can be easily extended to the case where several administrators exist at a system, each responsible for the relationship with a specific federation.

- *Global* objects. These are objects created directly in the federation by the federation administrator or by users explicitly authorized for that.
- *Imported* objects. These are objects defined and stored at some system taking part in the federation and imported "as-is" by the federation.
- *Composite* objects. These are objects defined on other global or imported objects (e.g., by aggregation).

3.2 Notations

Before proceeding further to illustrate our model we introduce some notations that will be used throughout the paper. In the following, we will use letters U, G, O, and A to denote a set of users, groups, objects, and authorizations, respectively. Subscripts will be used to denote the system, or federation, to which they refer. For instance, O_s denotes the set of objects at system s and G_f denotes the set of groups defined at federation f.

To take distribution into consideration we suppose each user/group identifier has associated the system at which it has been defined. User identifiers at system s will therefore have the form user@s. Analogously, subject (i.e., user or group) identifiers at a federation f will have the form subject@f. We will omit the specification of the system when it is not needed in the explanation or it is clear from the context. For instance, the indication of the system in the subject of an authorization will be omitted when the system is the same as the one at which the authorization is stored.

Subject patterns can appear instead of specific subjects in authorizations. Subject patterns are be specified by using the wild character in place of a specific identifier. We consider two kinds of patterns. Pattern "$*@s$", which indicates any identifier at system s, and pattern "$*$", which indicates any identifier at any system. In the following, notation U_s indicates the set containing all the users' identifiers at s together with $*@s$. Given a collection $s_1, \ldots, s_n$ of systems participating as customers in a federation f, we use notation $U_{customers(f)}$ to indicate the set $U_{s_1} \cup \ldots \cup U_{s_n} \cup \{*\}$. A pattern covers all the identifiers to which it can be reduced by appropriately instantiating the wild character. We say that a patterns p *covers* an identifier u@s, denoted as $p \triangleright \texttt{u@s}$, if either *i)* $p = \texttt{u@s}$, *ii)* $p = *\texttt{@s}$, or *iii)* $p = *$.

3.3 Working of the system

Users of systems registered as customers of a federation and wishing to access the federation must explicitly open a working session by connecting to the federated system (see Figure 1).

Connection requires identification of the user (1) and corresponding authentication of his/her identity by the federation. This identity will be used by the federation for enforcing access control. (Note that this assumption does not rule out the possibility of anonymous connection. Anonymous connection may be treated with a special user's identifier, e.g., anonymous). Besides the identity with which (s)he connected to the federation, a user has also associated an attribute containing his/her identity at the local

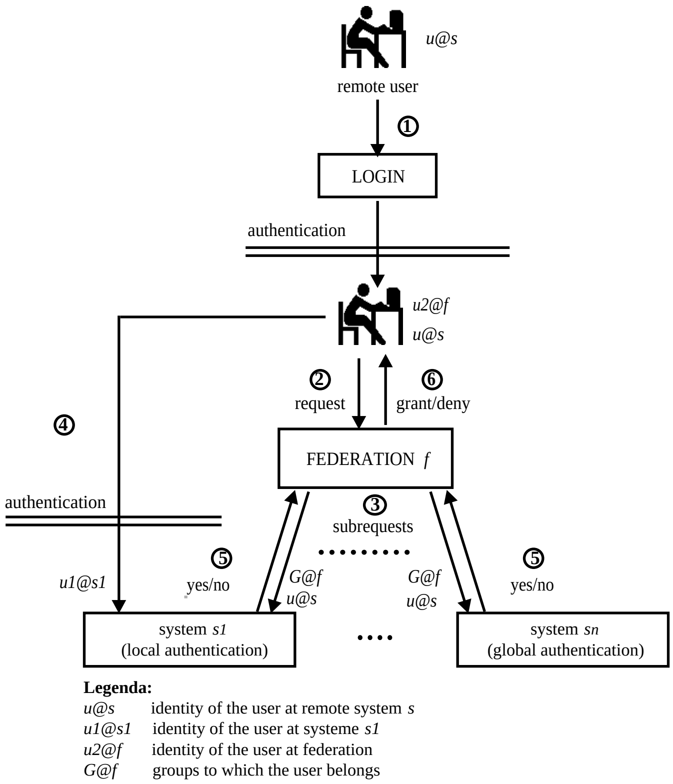

Figure 1: Working of the system

system from which he connected to the federation.[3] Once the user has been authenticated by the federation, s/he can submit requests to access federated objects (2). Accessing a federated object may require to access the corresponding underlying local objects stored at the participating systems. Each user request must therefore be split, for both access control and data retrieval, into a set of requests on the underlying local objects (or a subset of them). Then, the federation sends an access request for the groups to which the user belongs together with the remote identity of the user to each system storing a local object involved in the transaction (3). In case of local authentication the user will need to identify him/herself and be authenticated at the local system (4). Then, each local system controls whether the access request is allowed and returns the result to the federation (5). The federation will then take the access decision (6) on the basis of the replies of the different systems. We will illustrate access control more in details in Section 6.

[3]To allow non disclosure of identities, the case of attribute only partially specified, i.e., with indication only of the system, may be considered.

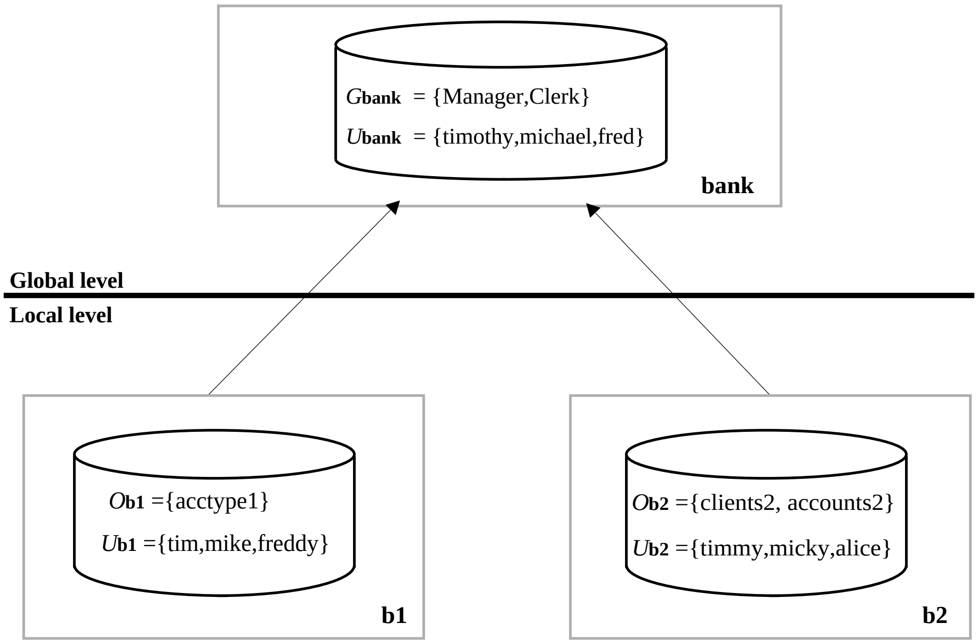

Figure 2: An example of federation

3.4 Running example

In the remainder of this paper we refer our examples to a federation connecting different branches of a bank. The federation is composed of a federated system `bank` and two component systems (branches) `b1` and `b2`. Objects and users at the branches, illustrated in Figure 2, are as follows:

- $O_{\mathtt{b1}} = \{\mathtt{acctype1}\}$

  `acctype1` contains conditions regulating accounts (e.g., interest rate, minimum balance required).

- $O_{\mathtt{b2}} = \{\mathtt{clients2,accounts2}\}$

  `clients2` contains information on `b2`'s clients.

  `accounts2` contains information about accounts opened at `b2`.

- $U_{\mathtt{b1}} = \{\mathtt{tim,mike,freddy}\}$

- $U_{\mathtt{b2}} = \{\mathtt{timmy,micky,alice}\}$

We also consider the following users and groups are defined at the federated system:

- $U_{\mathtt{bank}} = \{\mathtt{timothy,michael,fred}\}$

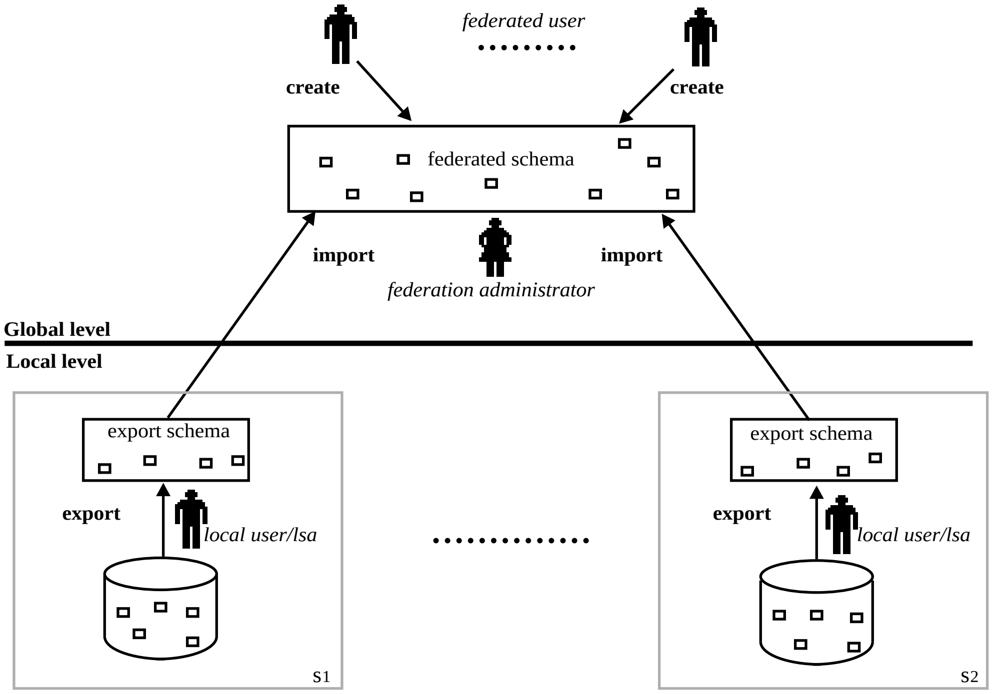

Figure 3: Populating the federation

- $G_{\texttt{bank}} = \{\texttt{Manager},\texttt{Clerk}\}$, where $\texttt{Manager} = \{\texttt{michael},\texttt{fred}\}$ and $\texttt{Clerk} = \{\texttt{timothy},\texttt{michael}\}$

4 Population of the federation

The federation can be populated directly, by explicitly creating global objects in the federation, or indirectly, by importing objects from the component systems (see Figure 3). In the remainder of this section we discuss the authorizations required and the operations that can be executed to populate the federation.

4.1 Authorizations for the population

Global objects can be created by any user explicitly authorized for that. Authorizations allowing users to create objects in the federation, which are specified by the federation administrator/s, have the following form.

Definition 1 (Create authorization) Let f be a federation, a create authorization $a \in A_f$ is a triple of the form $\langle \textit{fed-subject}, \texttt{create}, \textit{remote-id} \rangle$, with $\textit{fed-subject} \in U_f \cup G_f$ and $\textit{remote-id} \in U_{customers(f)}$.

Authorization $\langle \textit{fed-subject}, \texttt{create}, \textit{remote-id} \rangle$ states that federated user *fed-subject* (its member in case *fed-subject* is a group) connected to the federation through some customer system with a remote login covered by *remote-id* can create objects in the federation.

Example 1 Consider the following authorizations in $A_{\texttt{bank}}$.
⟨`fred`,`create`,$*$⟩
⟨`*`,`create`,`tim@b1`⟩
⟨`Manager`,`create`,$*$@`b2`⟩
These authorizations permit the creation of objects in `bank` to: federated user `fred` (first authorization); remote user `tim@b1` (second authorization); and all users members of group `Manager` which are connected to the federation from system `b2` (third authorization).

□

Population of the federation by getting objects from provider systems requires agreement of both the local object's administrator on one side and the federation administrator on the other side. At the local system the objects administrators must be willing to export their objects to the federation. At the federated system, the federation administrator must be willing to import the objects in the federation. *We allow objects to be exported with reference to specific access modes.* For instance, a user can decide to export one of his/her objects (i.e., allow it to be accessed by the federated users) only for reading and another one of his/her objects for both reading and writing.

In order to export their objects, users must be explicitly authorized for that by the system administrator. Export authorizations are defined as follows.

Definition 2 (Export authorization) Let s be a system. An export authorization $a \in A_s$ is a triple of the form ⟨*loc-user*,`export`,f⟩ with *loc-user* $\in U_s$ and $s \in$ *providers(f)*.

Authorization ⟨*loc-user*,`export`,f⟩ states that user *loc-user* can export the objects s/he administers to federation f.

Users can also delegate the system administrator to export their objects, possibly with reference to specific access modes. Reasons for delegation can be various. On the one hand, users may not want to worry about federations in which the system participates and about specifying authorizations for users of the federation. On the other hand, the administrator him/herself may wish to not allow direct export of objects by users, thus retaining control on what the system exports (a kind of centralized administration). Authorizations that allow system administrators to export objects are as follows.

Definition 3 (Del_export authorization) Let s be a system. An export delegation authorization $a \in A_s$ is a 4-tuple ⟨`lsa`,`del_export`,*loc-object*,*modes*⟩ where `lsa` denotes the system administrator, *loc-object* $\in O_s$ is the object whose export is delegated, and *modes* $\subseteq M_{loc\text{-}object}$ is the set of access modes that can be exported.

Authorization ⟨`lsa`,`del_export`,*loc-object*,*modes*⟩ $\in A_s$ states that the administrator of system s can export object *loc-object* with access modes *modes* to any federations for which the system is registered as provider. (Note that, since the administrator is unique, its indication could have been omitted. However, to make the model easily extendable to the case of multiple administrators we prefer to explicitly indicate it.)

Example 2 Authorization ⟨`lsa`,`del_export`,`acctype1`,{`read`}⟩ $\in A_{\texttt{b1}}$ states that the local administrator of `b1` can make object `acctype1` available for reading to all the federations in which branch `b1` participates.
Authorization ⟨`alice`,`export`,`bank`⟩ $\in A_{\texttt{b2}}$ states that user `alice@b2` can make the objects she administers available to federation `bank`. □

4.2 Export and import schema specification

Users authorized for that can export objects, possibly with reference to specific access modes, to a federation. When exporting an object, users must also decide the administrative policy establishing who can specify authorizations to access the object once imported in the federation. Our model supports three different policies:

- *system retained* **(SR)** Access authorizations can be specified only by the local administrators of the object.
- *federation controlled* **(FC)** The object is freely available to the federation. Access authorizations regulating access are specified only by the federation administrator.
- *cooperative* **(C)** Access authorizations can be granted by both the local administrator and the federation administrator.

The set of objects exported by a given system s to a federation f together with the indication of the specific modes exported and the administrative policy to be applied is called the *export schema security specifications* of s for f, as stated by the following definition.

Definition 4 (Export Schema security specifications) Let s be a system and f be a federation. The *Export Schema security specifications* of s for f, denoted with $ES_{s,f}$, is a set composed of 4-tuples of the form $\langle o, M, p, e\rangle$ where $o \in O_s$, $M \subseteq M_o$, $p \in \{\mathbf{SR}, \mathbf{FC}, \mathbf{C}\}$, and $e \in U_s$.

Tuple $\langle o, M, p, e\rangle \in ES_{s,f}$ indicates that user e has exported object o for access modes M under policy p.

As we have already discussed, actual inclusion of objects in the federation requires their import by the federation administrator. The federation administrator can import objects from export schemas defined at any customer system. Objects can be imported with reference to specific access modes. At the global level a *secure federated schema* is maintained which states the objects part of the federated data, the access modes executable on them, and the administrative policy to which they are subject.

Definition 5 (Secure Federated Schema) Let f be a federation. The *Secure Federated Schema* of f, denoted with FS_f, is a set of 4-tuples of the form $\langle o, M, p, c\rangle$ where $o \in O_f$, $M \subseteq M_o$, $p \in \{\mathbf{SR}, \mathbf{FC}, \mathbf{C}, \mathbf{G}, \mathbf{U}\}$ and $c \in U_f \cup \texttt{fed-adm}$.

Tuple $\langle o, M, p, c\rangle$ indicates: that object o has been inserted in the federated schema by user c, that the object is accessible with access modes M, and that it is subject to administrative policy p. User c is the federation administrator `fed-adm` in case of imported objects; it is the creator of the object in case of global or composite objects.

Secure federated schema is defined by the system on the basis of the import operations executed by the administrator and the create and define operations executed by users. The create operation allows users to create new objects. The define operation allows users to define composite objects. In case of imported objects, M and p are exactly those indicated in the export schema. In case of global objects, p has value `global` (**G**) and M is equal to all the access modes that can be exercised according to the object's type. In case of composite objects, M is the set of all the access modes that can be executed on all the component objects. The administrative policy is the one specified for the component objects if all the component objects have the same administrative policy for the corresponding local access mode. It has value `undefined` (**U**) otherwise. We assume that composite objects are always defined on objects present in the federated schema. This implies that an object must be imported before being used in the definition of a composite object.

The secure federated schema must be defined consistently with the export schema security specifications. Specification consistency is formalized by the following definition.

Definition 6 (Consistent specifications) Let f be a federation and $\langle o, M, p, c\rangle \in FS_f$ a specification in it. $\langle o, M, p, c\rangle$ is *consistent* if it satisfies any of the following conditions:

- o is a global object, $p = \mathbf{G}$, and $M \subseteq M_o$
- o is an imported object corresponding to object $\hat{o}$ at system $s \in customer(f)$ and $\exists\langle o', M', p', e'\rangle \in ES_{s,f}$, $o' = \hat{o}$, $M \subseteq M'$, $p = p'$.
- o is a composite object defined on $o_1, \ldots, o_n$ and $\exists\langle o_i, M_i, p_i, c_i\rangle \in FS_f, M \subseteq M_i, i = 1, \ldots, n$. Moreover, $p = p_1$ if $p_1 = p_2 = \ldots = p_n$; $p = \mathbf{U}$ otherwise.

Definition 7 (Federated schema consistency) A federated schema is *consistent* if and only if each 4-tuple $\langle o, M, p, c\rangle \in FS_f$ is consistent.

Table 2 summarizes the operations for defining the export schema security specifications and secure federated schema. Note that each operation will be successfully executed only if there exists an authorization for it. Note also that the syntax is restricted to the consideration of information related to the security specifications. Data definition and management is outside the scope of this paper.

The execution of the export operation for access modes *modes* on object *loc-object* with administrative policy *adm-policy* results in the addition of 4-tuple $\langle$*loc-object*,*modes*,*adm-policy*,*exporter*$\rangle$ in $ES_{s,f}$, where *exporter* is the user who required the operation. This tuple will be removed upon execution of a withdraw operation. Analogously, execution of an import, define, or create operation results in the insertion of the corresponding 4-tuple in the federated schema. Insertion of the 4-tuple must maintain the federated schema consistency.

Operation		Semantics
At the local level	EXPORT *loc-object* IN *f* WITH PRIVILEGES *modes* AND ADMINISTRATION *adm-policy*	To declare the object exportable from a system for privileges *modes*
	WITHDRAW *loc-object* FROM *f*	To remove *loc-object* from the export schema for the federation
At the global level	IMPORT *loc-object* FROM *s* AS *fed-object*	To import the object in the federation
	DEFINE *fed-object* ON *object-list*	To define a composite object
	CREATE *fed-object*	To create a global object

Table 2: Operations for defining export and federated schema

Note that the withdraw operation, which removes objects from export schemas, may affect the federated schema consistency. Although we require the federated schema to be consistent, i.e., require that changes to the export schema be propagated to the federated schema, we note that more lazy approaches similar to those illustrated in Section 2.6 for maintaining authorization consistency can be applied.

Example 3 Consider the federation of Figure 2 and the export and del-export authorizations of Example 2. Let `fed-adm@bank` be the federation administrator. Consider now the following operations are executed:

- **At branch** `b1`:

EXPORT `acctype1` IN `bank` WITH PRIVILEGES {`read`} AND ADMINISTRATION **FC**

- **At branch** `b2`:

EXPORT `accounts2` IN `bank` WITH PRIVILEGES {`read`} AND ADMINISTRATION **SR**
EXPORT `clients2` IN `bank` WITH PRIVILEGES {`read,write`} AND ADMINISTRATION **FC**

- **At federation** `bank`:

IMPORT `acctype1` FROM `b1` AS `acctype`
IMPORT `accounts2` FROM `b2` AS `accounts`
IMPORT `clients2` FROM `b2` AS `clients2`
DEFINE `acctypeclients` ON {`acctype,clients2`}

The resulting export schema security specifications and secure federated schema are illustrated in Figure 4. □

During the life time of a federation, authorizations for exporting objects may change, i.e., new authorizations be granted and existing authorizations revoked. Revocation introduces the problem of dealing with objects whose export was authorized by the authorization being revoked. In this paper we do not deal with the problem of enforcing

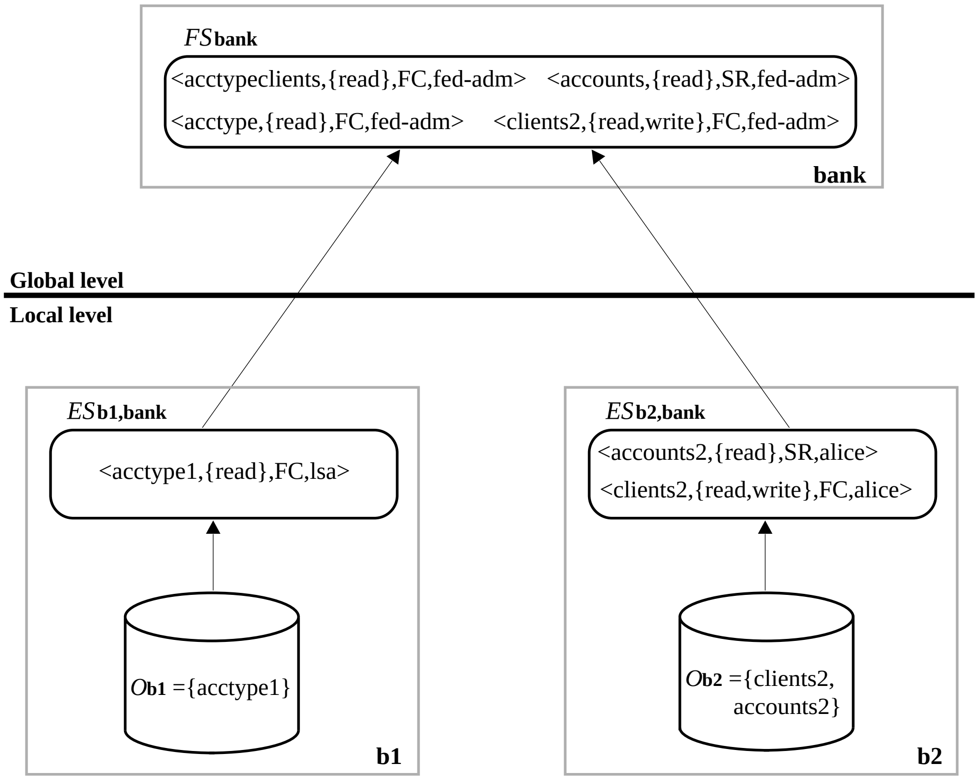

Figure 4: Example of export and federated schema

revocation of authorizations. However, we note that two different strategies, one *destructive* (i.e., with object withdrawal) and one more *conservative* (i.e., without object withdrawal), can be adopted with respect to this. In the destructive strategy whenever a user is revoked the export authorization for a federation, all the objects exported by the user to that federation are removed from the export schema. Analogously, if the system administrator is revoked the del-export privilege on an object, the object will be removed from the export schemas in which it was included by the system administrator. In the conservative strategy, upon revocation of an export/del-export authorization only this authorization is deleted and the export schema remains unchanged.

5 Access authorizations

In our model we allow the specification of authorizations at both the global level, on the federated objects, and at the local level, on the objects stored at the local systems.

Authorizations at the global level can be specified only by the federation administrator in case of imported objects. They can be specified by the object's creator in case of global or composite objects.

Authorizations on federated objects can be granted to single users or to groups of them. Authorizations can also contain restrictions on the remote user's identity or on the remote system from which the user is connected. Subjects and remote identities can be specified either by using specific identifiers or by using subject patterns. Authorizations at the global level are defined as follows.

Definition 8 (Global access authorization) Let f be a federation, an access authorization $a \in A_f$ is a 4-tuple $\langle$*fed-subject*,*mode*,*fed-object*,*remote-id*$\rangle$ where: *fed-subject* $\in U_f \cup G_f$, *mode* $\in M_{\textit{fed-object}}$, *fed-object* $\in O_f$, *remote-id* $\in U_{\textit{customers}(f)}$.

Authorization $\langle$*fed-subject*,*mode*,*fed-object*,*remote-id*$\rangle$ states that *fed-subject* (any of its members in case it is a group), connected to federation f through some customer system with a login covered by *remote-id*, can exercise privilege *mode* on *fed-object*.

Example 4 Consider the following authorizations in $A_{\texttt{bank}}$.
$\langle$`timothy`,`read`,`acctype`,`*`$\rangle$
$\langle$`*`,`read`,`accounts`,`tim@b1`$\rangle$
$\langle$`*`,`read`,`acctype`,`*@b2`$\rangle$
$\langle$`Manager`,`read`,`acctypeclients`,`*`$\rangle$}
The first authorization states that federated user `timothy` can read `acctype`. The second authorization states that remote user `tim@b1` can read `accounts`. The third authorization states that all users connected to the federation from `b2` can read `acctype`. The fourth authorization states that members of group `Manager` can read `acctypeclients`. $\square$

At the local level, users can specify access authorizations on the objects they have exported. Subjects of authorizations at the local level are groups defined at the federation. The reason why groups are considered for authorizations at the local level is to avoid requiring each system to be informed of identities of single users registered at the federation. Indeed, although it is reasonable to assume some knowledge about global identifiers, it is quite improbable that local systems can retain track of identifiers of each single user of the federation. The consideration of groups overcome this problem. Each federation declares a set of groups into which federated users are organized and for which users at local systems can specify authorizations. Changes to groups (i.e., addition or removal of members) will not therefore have any effect on authorizations specified at the local systems. (One may object that, by specifying authorizations for groups they do not manager, users may lose control on whom they give access. To avoid this we allow, at the local level, the indication of specific identity in authorizations and the specification of negative authorizations.)

Authorizations at the local level can also contain constraints on the specific identity of the users requiring the access. In case of global authentication, the constraints will refer to the remote user's identity at the system from which the user connected to the federation. In case of local authentication, the constraints will refer to the local identity of the user as authenticated by the system.

Access authorizations at local level can be positive or negative. Positive authorizations state accesses that must be granted, whereas negative authorizations state accesses that must be denied. Possible conflicts are solved according to the denials take precedence policy: if both a positive and a negative authorization are specified for a same access, the access is denied.

Local authorizations are defined as follows.

Definition 9 (Local access authorization) Let s be a system in a federation f, an access authorization $a \in A_s$ is a 5-tuple $\langle$*fed-group*,*mode*,*sign*,*loc-object*,*id*$\rangle$ with *fed-group* $\in G_f$, *mode* $\in M_{loc\text{-}object}$, *sign* $\in \{$+,-$\}$, *id* $\in U_s \cup U_{customers(f)}$.

Authorization $\langle$*fed-group*,*mode*,*sign*,*loc-object*,*id*$\rangle$ states that all members of *fed-group*, with a login (remote if authentication is global, and local otherwise) covered by *id*, can (if *sign* = +) or cannot (if *sign* = -) exercise privilege *mode* on *loc-object*.

Example 5 Consider the following authorizations defined at each branch of federation bank.

$A_{\texttt{b1}} = \{\langle$*,read,-,acctype1,*@b2$\rangle\}$
$A_{\texttt{b2}} = \{\langle$Clerk@bank,read,+,clients2,*$\rangle$
$\langle$*,read,-,accounts2,timmy$\rangle$
$\langle$Clerk@bank,read,+,accounts2,*$\rangle\}$

The first authorization specified at b1 states that all users connected to the federation from b2 cannot read acctype1.
The first authorization specified at b2 states that the members of group Clerk can read clients2.
The second authorization states that user timmy@b2 cannot read accounts2.
The third authorization states that members of group Clerk can read accounts2. □

The reason for considering both positive and negative authorizations at the local level is to give exporters *a means of retaining control on the accesses allowed on their objects.* This feature is very important for two main reasons. First, in our model authorizations are specified for groups of users. However, an exporter may wish to grant a whole group an access but at the same time make sure that a specific user will not be able to exercise it. Since user's groups are defined at the federation system, and therefore the exporter has no means of controlling their configuration (for instance by excluding the specific user), the specification of negative authorizations may be the only means to express this requirement. Second, in the case of federation controlled administration, the exporter delegates the federation administrator the task of specifying access authorizations on his object once imported. This implies that federated users will not need to have local privileges in order to access the object. However, by specifying negative authorizations the exporter can restrict the accesses that federated users can exercise on the object. As a matter of fact no access, even if authorized by the administrator, will be allowed if there exists a negative authorization for it at the local level.

Subject patterns, together with positive and negative authorizations, can be used to express different protection requirements, as illustrated by the following example.

Example 6 Consider the federation defined in Example 3. Object `acctype1` is exported under the federation controlled policy. Hence, in order to access the object, users need to have the authorization for the access at the global level. By contrast, no authorization is necessary at the local level. Suppose that the local system administrator *lsa*@`b1` wants to make sure that users from `b2` will not be able to read the object. He can do so by specifying negative authorization $\langle *,$`read`$,-,$`acctype1`$,$`*@b2`$\rangle$.

Consider now object `accounts2` exported under the site retained policy. Suppose that `alice` wishes to grant the read access to group `Clerk`, but she does not want local user `timmy` to read the object, even if `timmy` is a member of `Clerk`. She can express this requirement by specifying authorizations: $\langle$`Clerk@bank`,`read`$,+,$`accounts2`$,*\rangle$ and $\langle *,$`read`$,-,$`accounts2`,`timmy`$\rangle$ in $A_{\texttt{b2}}$. □

Since authorizations can contain patterns, a single authorization can be applied to more access requests. This concept is captured by the definition of covering authorization as follows.

Definition 10 (Covering authorization) A global authorization $a = \langle fs,m,fo,rid\rangle$ is said to *cover* a tuple $t = \langle u',m',o',id'\rangle$, written $a \triangleright t$, iff: $fs \triangleright u'$, $m = m'$, $fo = o'$, $rid \triangleright id'$.
A local authorization $a = \langle fs,m,sgn,fo,rid\rangle$ is said to *cover* a tuple $t = \langle u',m',sgn',o',id'\rangle$, written $a \triangleright t$, iff: $fs \triangleright u'$, $m = m'$, $sgn = sgn'$, $fo = o'$, $rid \triangleright id'$.

Intuitively, an authorization a cover a tuple $t = \langle u',m',o',id'\rangle$ corresponding to an access request, if each element of a covers the corresponding element of t. For instance, local authorization $\langle *,$`read`$,-,$`acctype1`,`*@b2`$\rangle \in A_{\texttt{b1}}$ covers tuples $\langle$`Clerk@bank`,`read`$,-,$`acctype1`,`timmy@b2`$\rangle$ and $\langle$`Manager@bank`,`read`$,-,$`acctype1`, `timmy@b2`$\rangle$. It does not cover tuple $\langle$`Clerk@bank`,`read`$,-,$`acctype1`,`mike@b1`$\rangle$.

6 Access control

Users connected to the federation can submit requests to access federated objects. In order to determine whether to grant or deny an access, authorizations specified at the federation and at the local systems involved in the access must be controlled. Specific controls and additional authentication processes required depend on: *i)* the type of object (global versus imported or composite), *ii)* the kind of administrative policy of the component object/s, and *iii)* the authentication policy required by each system involved. Each request on an imported or composite object is translated into a request, or a set of requests, on the corresponding local object/s. Each of these requests must be communicated to the appropriate system for both access control (since local authorizations must be checked to determine whether the access must be allowed or denied) and data retrieval (since the data stored at the different systems are not replicated at the federation). The mapping of access operations on a federated object onto access operations on the corresponding local objects is enforced by the data management system of the federation. It is outside the scope of this paper to discuss this mapping. We assume that

Input: Request (u, m, o, rid) of federated user u, remotely connected as rid, to exercise access mode m on federated object o
Output: The access control decision: *grant access* if the access is to be allowed; *deny access* otherwise.

At the federated system f

1. **If** $\exists\langle o, M, p, c\rangle \in FS_f$
   then $policy := p$ **else** *deny access*
2. Determine the groups GS to which the user belongs.
3. /* Access control at global level */
   If $policy = \mathbf{G}$
   then if $\exists a \in A_f$ such that $a \rhd \langle u, m, o, rid\rangle$ or $a \rhd \langle GS, m, o, rid\rangle$
   then *grant access*
   else *deny access*
   else if $policy \neq \mathbf{SR}$
   then if $\nexists a \in A_f, a \rhd \langle u, m, o, rid\rangle$ or $a \rhd \langle GS, m, o, rid\rangle$
   then *deny access*
4. /* Split into requests on component objects */
   Decompose access (m, o) into a set of accesses $S = \{(m_1, o_1), \ldots, (m_n, o_n)\}$ with $o_i,\ i = 1 \ldots n$ a global or an imported object.
5. /* Check all accesses for which component object is global */
   For each object o_i let p_i the policy such that $\langle o_i, M_i, p_i, e_i\rangle \in FS_f$
   if $\exists(m_i, o_i) \in S$ such that $p_i = \mathbf{G}$ and $\nexists a \in A_f,\ a \rhd \langle u, m_i, o_i, rid\rangle$ or $a \rhd \langle GS, m_i, o_i, rid\rangle$
   then *deny access*
   else if $\nexists(m_i, o_i) \in S$ such that $p_i \neq \mathbf{G}$
   then *grant access*
6. /* Send requests on imported objects to local system */
   For each pair $(m_i, o_i) \in S$ such that $p_i \neq \mathbf{G}$ **do**
   let o'_i be the local object corresponding to o_i
   send request $r_i = (GS, m_i, o'_i, rid)$ to the system from which o'_i was imported
7. /* Collect replies from system and return response to the user */
   Collect replies to all requests r_i
   If the reply has value `true` for all requests r_i **then** *grant access* **else** *deny access*

At each local system s
/* upon reception of request $r_i = (GS, m_i, o'_i, rid)$ from federation f */

1. **if** $\exists\langle o'_i, M, p', e\rangle \in ES_{s,f}$ with $m_i \in M$
   then $adm\text{-}policy := p'$
   else $reply :=$ `false`, go to step 4
2. **if** authentication is global
   then $id = rid$
   else Authenticate user. Assign id the local user identity.
3. **case** *adm-policy* **of**
   SR or **C**: **if** $\exists a \in A_s, a \rhd \langle GS, m_i, +, o'_i, id\rangle$ and $\nexists a \in A_s, a \rhd \langle GS, m_i, -, o'_i, id\rangle$
   then $reply :=$ `true`
   else $reply :=$ `false`
   FC: **if** $\nexists a \in A_s, a \rhd \langle GS, m_i, -, o'_i, id\rangle$
   then $reply :=$ `true`
   else $reply :=$ `false`
4. Send *reply* to f

Figure 5: Access control

each operation to access a federated object o with access mode m is mapped by the data management system onto a set of operations $\langle m_1, o_1\rangle, \ldots, \langle m_k, o_k\rangle$, where $o_1, \ldots, o_k$ are local objects corresponding to o and $m_1, \ldots, m_k$ are the access modes to be exercised on these objects. Note that $k = 1$ in case of imported objects.

Figure 5 illustrates the different controls to be executed at the federated system and at each local system involved in an access. In the figure, given a set of groups GS, we use notation $\langle GS{,}m{,}o{,}rid\rangle$ as an abbreviation for the set $\{\langle g_i{,}m{,}o{,}rid\rangle \mid g_i \in GS\}$. Moreover we use $a \rhd \langle GS{,}m{,}o{,}rid\rangle$ as an abbreviation for $a \rhd \langle g_1{,}m{,}o{,}rid\rangle \vee a \rhd \langle g_2{,}m{,}o{,}rid\rangle \vee \ldots \vee a \rhd \langle g_k{,}m{,}o{,}rid\rangle$, where $g_1, \ldots, g_k$ are the groups in GS.

The algorithm works as follows. Let f be a federation and u be a federated user connected to the federation with some remote login rid. Consider the request to exercise access mode m on object o submitted to the federation by this user. If the object does not belong to the federated schema, then the access is denied (step 1). Otherwise, in step 2, the set of groups GS to which the user belongs is determined. Step 3 enforces access control against the global authorizations. If the object is global (administrative policy **G**) this is the only control that must be executed. Hence, the access is granted or authorized according to whether an authorization covering the request exists or not. Otherwise, the controls to be executed depend on the administrative policy specified for the object. If the administrative policy is system retained (**SR**), no authorization is needed for the access at the global level and therefore no authorization check is performed by the federation. In any other case (i.e., administrative policy equal to **FC**, **C**, or **U**), the system looks for an authorization covering the request. If no such authorization exists the access is denied, otherwise the process continues. Hence, in step 4, the access request on the federated object is split into the corresponding accesses on component objects. If any of these accesses is on a global object and no authorization exists for it, the request is denied. Otherwise, if all accesses to global objects are authorized and no access to local objects is required the request is granted (step 5). Otherwise, the algorithm proceeds by sending, for each access on an imported object, a request to the corresponding system for the set of groups GS with remote identity rid (step 6). The access is granted if all the accesses into which the request has been split are authorized; it is denied otherwise (step 7).

Access control at each local system s works as follows. Upon reception of a request by a federation, the export schema of the system for the federation is controlled. If there is no triple in the export schema for the object with a set of access modes containing the action required, a `false` reply is immediately returned to the federation. Otherwise the control proceeds. The identity to be considered in the access control depends on the authentication policy established for the system. If the authentication is global, the remote identity of the user as communicated by the federation is considered. If the authentication is local, the user will be asked to identify him/herself. This local identity, once authenticated, will be used in place of the remote identifier in the access control. Hence the local authorizations are checked. In case of system-retained or cooperative administrative policy a `true` is returned only if there exists a positive authorization and there does not exist any negative authorizations for the access. In case of `federation controlled` administrative policy a `true` is returned if no negative authorization for the

Constraints on authorizations		Administrative policy: System retained	Federation controlled	Cooperative	Global
At federation f (in A_f)			$\exists$a: a$\triangleright\langle$u,m,o,rid$\rangle$ or a$\triangleright\langle$GS,m,o,rid$\rangle$	$\exists$a: a$\triangleright\langle$u,m,o,rid$\rangle$ or a$\triangleright\langle$GS,m,o,rid$\rangle$	$\exists$a: a$\triangleright\langle$u,m,o,rid$\rangle$ or a$\triangleright\langle$GS,m,o,rid$\rangle$
At system s of o (in A_s)	global auth.	$\exists$a: a$\triangleright\langle$GS,m,+,o$'$,rid$\rangle$ $\nexists$a: a$\triangleright\langle$GS,m,−,o$'$,rid$\rangle$	$\nexists$a: a$\triangleright\langle$GS,m,−,o$'$,rid$\rangle$	$\exists$a : a $\triangleright$ $\langle$GS,m,+,o$'$,rid$\rangle$ $\nexists$a: a$\triangleright\langle$GS,m,−,o$'$,rid$\rangle$	
	local auth.	$\exists$a: a$\triangleright\langle$GS,m,+,o$'$,lid$\rangle$ $\nexists$a: a$\triangleright\langle$GS,m,−,o$'$,lid$\rangle$	$\nexists$a: a$\triangleright\langle$GS,m,−,o$'$,lid$\rangle$	$\exists$a: a$\triangleright\langle$GS,m,+,o$'$,lid$\rangle$ $\nexists$a: a$\triangleright\langle$GS,m,−,o$'$,lid$\rangle$	

Legend: o' = local object corresponding to o
rid = remote identity of user u
lid = identity of user u at system s

Table 3: Conditions necessary to grant request by user u of groups G_u with remote identity rid to exercise mode m on federated object o

access exists.

Table 3 summarizes the authorizations necessary at the global and the local level in order for a request on a global or imported object to be granted. In case of composite objects, at the local level the conditions must be satisfied with reference to each access (m_i, o_i) in which the global operation is split.

Example 7 Consider the federation defined in Example 3 and the authorizations specifications of Examples 4 and 5. The resulting authorizations and schemas are illustrated in Figure 6. Remember that: `acctype`, `clients2`, and `accounts` are imported objects corresponding to local objects `acctype1`, `clients2`, and `accounts2`, respectively and that `acctypeclients` is a composite object defined on `acctype` and `clients2`.

Suppose also that:

- `b1` enforces local authentication while `b2` enforces global authentication;
- `tim@b1` and `timmy@b2` are local identifiers of federated user `timothy`;
- `mike@b1` and `micky@b2` are local identifiers of federated user `michael`;

Consider now user `tim@b1`, connected to the federation as `timothy`, who submits a request to read `accounts`. Since the administrative policy of `accounts` is system retained no control is executed at the federation. Hence a request for {`Clerk`} (the group to which `timothy` belongs) to read `accounts2` is sent to system `b2`. Since `b2` enforces local authentication, `timothy` will need to identify himself, as `timmy@b2`. Access control for request ⟨`Clerk@bank`,`read`,`accounts2`,`timmy`⟩ returns false due to the negative authorization stored at `b2`. Therefore the access is denied.

Suppose now that `mike@b1` connects to the federation as `michael` and requests to read `acctypeclients`. Access control at the global level is successful due to the authorization for group `Manager`. Since the administrative policy is federation controlled no authorization is needed at local level, however, local control must be executed to determine whether there exists a negative authorization for the access. Hence, the global access request is split into two local requests on corresponding local objects. The first

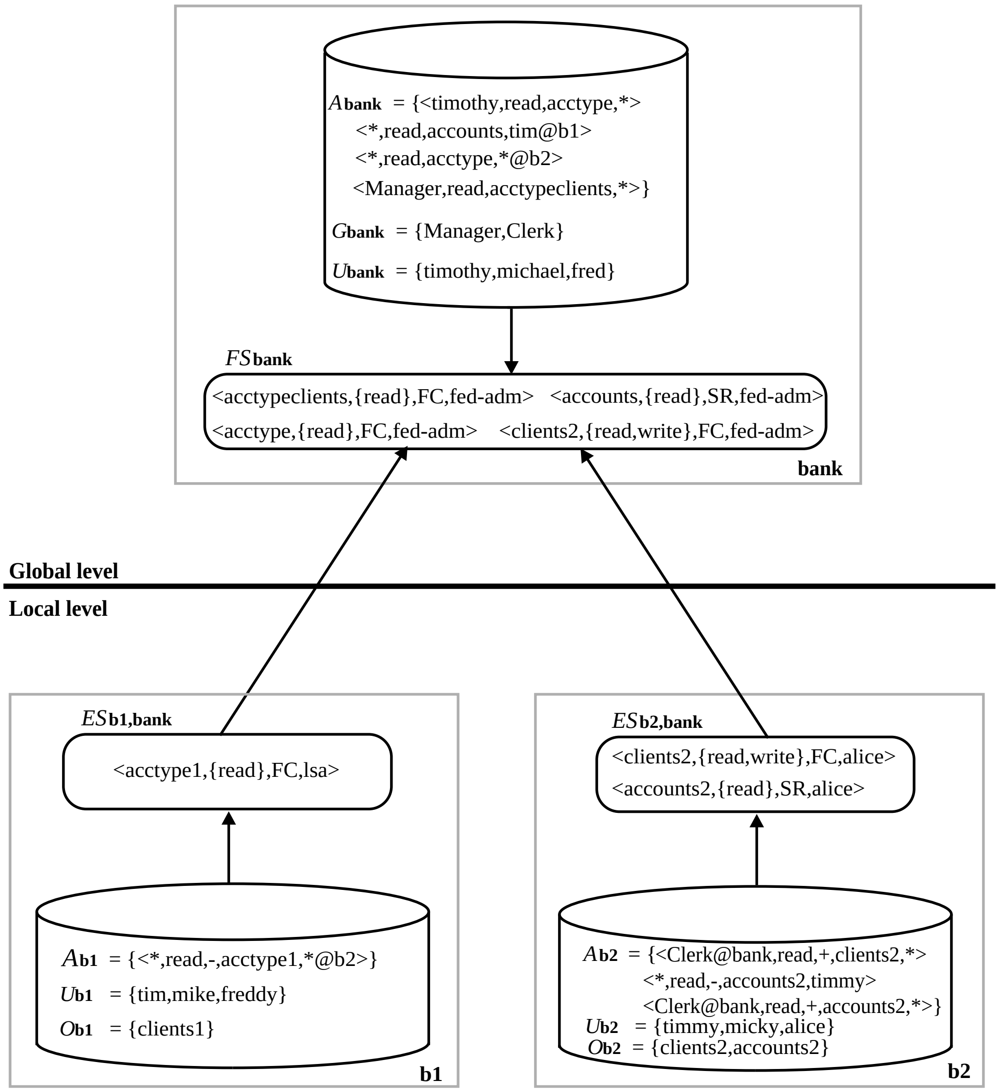

Figure 6: An example of federation

Security aspect		Wang and Spooner [30]	Mermaid [26]	Jonscher and Dittrich [14]	Our model
User's Authentication	federation	• requires users to explicitly identify themselves	• requires users to explicitly identify themselves	• requires users to explicitly identify themselves	• requires users to explicitly identify themselves
	local components	• global authentication	• local authentication	• local authentication • global authentication	• local authentication • global authentication
Access control	at the federation	• based on the global identity of the user	• based on the federated identity of the user	• based on the federated identity of the user	• based on the federated identity of the user • based on the remote identity of the user • based on the remote system's identity
	at each local component	• based on the global identity of the user	• based on the local identity of the user	• based on the local identity of the user (full or medium authorization autonomy) • based on the identity of the federation (low authorization autonomy)	• based on the federated identity of the user • based on the remote identity of the user • based on the remote system's identity • based on local identity of the user
Population of the federation			• import of objects	• import of objects	• direct creation • import of objects
Administration of authorizations for imported objects		• owner and users explicitly authorized	• cooperative	• cooperative • federation administrator	• local administrator • federation administrator • cooperative
Specification of authorizations		• top down derivation (with or without confirmation of loc. administrator)	• global and local authorizations are independent	• top down derivation	• global and local authorizations are independent
Global and local authorization consistency		• not required	• not required	• access time	• not required

Table 4: Approaches proposed to the solution of the problems discussed in Section 2

request for the set of groups {`Clerk,Manager`} with remote login `mike@b1` to access `acctype1` is sent to system `b1`. Since `b1` enforces global authentication, no further authentication is necessary. Since no negative authorization denying the access exists (the only negative authorization applies only to user remotely connected from `b2`), `b1` sends back a `true` reply, accepting the local requests. The second request for the set of groups {`Clerk,Manager`} with remote login `mike@b1` to access `clients2` is sent to system `b2`. Since `b2` enforces local authentication, `michael` needs to identify himself, as `micky@b2`. Since no negative authorization exists for the access, a `true` reply is returned by `b2`. Since both the local requests have been accepted the access is granted

□

7 Related work

Other authors have proposed approaches to regulate access in federated database systems. In this section we illustrate how some of these approaches have addressed the issues discussed in Section 2 and compare them with the approach proposed in this paper. Table 4 summarizes the discussion below.

The work of Wang and Spooner [30], although not explicitly addressed to federated databases, considers issues related to system autonomy for access control. The global database, corresponding to our federated schema, is obtained by integrating the local databases. Since a generic distributed, not federated, database is considered, there is no notion of export/import operations or of federated system. At each system a Global Functional Module (GFM) is responsible for interfacing the users and the global database in order to hide distribution. No distinction is made between local or global identities of users: a user has a single identity, we will call it global, which identifies him/her at each system. Authorizations to access global objects are specified at the global level. Whenever an authorization is granted at the global level the corresponding necessary authorizations must be granted at the local level (top down derivation). This process can be performed automatically (if the local system applies decentralized authorization) or through the mediation of the local system administrator (if the local system applies centralized authorization). Upon each access request by a user to access global data submitted at a system, the GFM stored at the system authenticates the user and decomposes his/her request into subrequests to be submitted at the different sites. The subject of the subrequests may be different in case of views, which are treated in a special way in the model.

In Mermaid [26], a front-end system to integrate multiple homogeneous relational DBMSs, an authorization model enforcing access control at both global and local level is presented. Local systems declare the objects that wish to share with Mermaid by defining views on the local databases. These views are then supplied to the builder and a single federated schema is developed. Access authorizations are independently specified both at the global level, in the Mermaid system, and at the local level, at each system. In order to use Mermaid a user must be registered for it. Access control at the global level is performed with respect to the identity with which the user is registered with Mermaid. Access control at the local level is performed with respect to the local identity of the users. To avoid users to have to identify themselves at each local system, Mermaid maintains a mapping between global and local identifiers together with an encrypted copy of the local password of the users. In this way, identification and proofs of identity of the users can be supplied to the local systems directly by Mermaid. With reference to our terminology, Mermaid uses cooperative administration and local authentication. Federation controlled administration together with global authentication can be enforced in Mermaid by associating to all the users a same local login, `mermaid`. Authorizations granted to this login will be therefore usable by all the users of the federation.

The model proposed by Jonscher and Dittrich in [14] supports three levels of local autonomy: *low authorization autonomy*, *medium authorization autonomy*, and *full authorization autonomy*. Intuitively: low authorization autonomy corresponds to the combination of our global authentication option with federation controlled administration; medium authorization autonomy corresponds to the combination of our global authentication option with cooperative administration; full authorization autonomy corresponds to the combination of our local authentication option with cooperative administration of authorizations. Authorizations, which can be positive or negative, are specified at the global level. The corresponding local authorizations are then derived at the local level (top-down derivation). The subject of the local authorizations is determined on the basis of mapping, specified by the global security administrator, establishing the correspondence between local and global identifiers. The grantor of an authorization at the global level can require consistency of the authorizations. In this case, the global authorization is inserted only if all the corresponding local grants can be enforced. Changes to local authorizations, which may affect consistency, are discovered and propagated at the time of access control.

Like our model, [26] and [14] allow to support different authentication options and administrative policies. However, in these models not all possible combinations of authentication options and administrative policies are supported. Moreover, all objects stored at a given system are subject to the same administrative policy. It is therefore not possible to enforce fine-grained specification of policies like in our model.

Other proposals have been made to control access in federated systems which, however differ from our work and the proposals above since they consider a loosely coupled architecture, where no notion of federated schema is provided [10, 16, 3] or they consider mandatory policies [18, 21, 28, 29].

8 Conclusions and future research

We have presented an authorization model for the protection of information in a tightly coupled federated system. The model allows users to decide which of their objects to share with the federation and how to share them. In particular they can specify the specific access modes allowed by federated users and the administrative policy establishing how authorizations on objects are to be specified. Inclusion of objects in the federated schema is a two step-process requiring agreement of both the administrator of the object at one side and the administrator of the federation at the other. Authorizations can be specified both at the global level, on federated objects, and at the local level on local objects exported to the federation. We have also illustrated the controls to be executed at the federation and at each local system to determine whether to grant or deny access to federated data. The model results flexible and able to support different kinds of protection requirements users may have.

Our paper leaves space for further work. A first issue we plan to investigate is the relationship between global and local authorizations. In the paper we have assumed that global and local authorizations are independently specified. The approach could be extended to the consideration of different strategies for the specification of authorizations. A first possible strategy is that upon specification of a global authorization, the local authorizations needed for the access globally granted to be allowed are derived and inserted at the local systems. Another strategy could be that of deriving global authorizations on federated objects from the authorizations specified locally. A further issue to be investigated concerns the communication of identities or group memberships between the federation and the systems. In this paper we have assumed that local access control on requests by a federated user is always carried out with reference to the groups to which the user belongs, and that the federation communicates these groups to the local systems. Alternative approaches can be considered such as the use of a call back mechanism, where a system explicitly asks whether a user belongs to a group, or the use of credentials [12]. Moreover, our model could be made more precise by considering specific data model at the global level . In this case, all the issues related to the translation of the export schemas into this data model should be taken into consideration. Finally, a further issue that can be investigated concerns the consideration of component systems with heterogeneous data models or heterogeneous security policies [11], and of different cooperation rules that can be established among systems to regulate access to the shared data.

References

[1] M. Abadi, M. Burrow, B. Lampson, and G. Plotkin. A Calculus for Access Control in Distributed Systems. Technical Report 70, DEC, System Research Center, Palo Alto, February 1991.

[2] P. Bernstein, V. Hadzilacos, and N. Goodman. *Concurrency Control and Recovery in Database Systems.* Addison-Wesley, 1988.

[3] B. T. Blaustein, C. D. McCollum, A. Rosenthal, and K. P. Smith. Autonomy and Confidentiality: Secure Federated Data Management. In *Proceeding of the 2nd International Workshop on Next Generation Information Technologies and Systems*, Naharia, Israel, June 1995.

[4] S. Castano. An Approach to Deriving Global Authorizations in Federated Database System. In *Proc. IFIP WG11.3 Working Conference on Database Security*, Como, Italy, July 1996.

[5] S. Castano, M.G. Fugini, G. Martella, and P. Samarati. *Database Security.* Addison-Wesley, 1995.

[6] S. Ceri and G. Pelagatti. *Distributed Databases–Principles and Systems.* McGraw-Hill, New York, 1984.

[7] S. De Capitani di Vimercati and P. Samarati. An Authorization Model for Federated Systems. In *Proc. of the Fourth European Symposium on Research in Computer Security (ESORICS'96)*, Rome, Italy, September 1996.

[8] S. De Capitani di Vimercati and P. Samarati. Access Control in Federated Systems. In *Proc. of the ACM SIGSAC New Security Paradigms Workshop*, Lake Arrowhead, CA, August 1996

[9] M. L. Goyal and G. V. Singh. Access Control in Distributed Heterogeneous Database Management Systems. *Computers & Security*, 10:661–669, 1991.

[10] D. Heimbigner and D. McLeod. A Federated Architecture for Information Management. *ACM Transactions on Office Information Systems*, 3(3):253–278, 1985.

[11] S. Jajodia, P. Samarati, V. S. Subrahmanian, and E. Bertino. A Unified Framework for Enforcing Multiple Access Control Policies. In *Proc. of the 1997 ACM International Sigmod Conference on Management of Data*, Tucson, AZ, May 1997.

[12] V. E. Jones, N. Ching, and M. Winslett. Credentials for Privacy and Interoperation. In *Proc. New Security Paradigms Workshop*, pages 93–100, La Jolla, California, U.S.A, August 1995.

[13] D. Jonscher and K. R. Dittrich. Access Control for Database Federations a discussion of the state-of-the-art. In *Proceeding DBTA Workshop on Interoperability of DBSs and DB Applications*, October 1993.

[14] D. Jonscher and K. R. Dittrich. An Approach for Building Secure Database Federations. In *Proceedings of the 20th VLDB Conference, Santiago, Chile*, 1994.

[15] D. Jonscher and K. R. Dittrich. Argos — A Configurable Access Control Subsystem Which Can Propagate Access Rights. In *Proc. 9th IFIP Working Conference on Database Security*, Rensselaerville, New York, U.S.A, August 1995.

[16] W. Kim, N. Ballou, J. F. Garza, and D. Woelk. A Distributed Object-Oriented Database System Supporting Shared and Private Databases. *ACM Transactions on Office Information Systems*, 9(1):31–51, January 1991.

[17] W. Litwin, L. Mark, and N. Roussopoulos. Interoperability of Multiple Autonomous Databases. *ACM Computing Surveys*, 22(3):267–293, 1990.

[18] J. McHugh and B. Thuraisingham. Multilevel Security Issues in Distributed Database Management Systems. *Computers & Security*, 7:387–396, 1988.

[19] M. Morgenstern, T. F. Lunt, B. Thuraisingham, and D. L. Spooner. Security Issues in Federated Database Systems: Panel Contributions. In C. E. Landwehr and S. Jajodia, editors, *Database Security, V:Status and Prospects, IFIP*, pages 131–148, 1992.

[20] B. C. Neuman and T. Ts'o. Kerberos: An Authentication Service for Computer Networks. *IEEE Communications Magazine*, 32(9):33–38, 1994.

[21] M. S. Olivier. A Multilevel Secure Federated Database. In *Proc. 9th IFIP Working Conference on Database Security, Rensselaerville*, pages 23–38, New York, U.S.A, August 1995.

[22] P. Samarati, E. Bertino, and S. Jajodia. An Authorization Model for a Distributed Hypertext System. *IEEE Transactions on Knowledge and Data Engineering*, 8(4):555–562, August 1996.

[23] R.S. Sandhu and P. Samarati. Authentication, Access Control and Intrusion Detection. In *CRC Handbook of Computer Science and Engineering* , A. Tucker (ed.), CRC Press Inc., pages 1929–1948, 1997.

[24] M. Satyanarayanan. Integrating Security in a Large Distributed System. *ACM Transactions on Computer Systems*, 7(3):247–280, August 1989.

[25] A. P. Sheth and J. A. Larson. Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. *ACM Computing Surveys*, 22(3):183–236, 1990.

[26] M. Templeton, E. Lund, and P. Ward. Pragmatics of Access Control in Mermaid. In *IEEE-CS TC Data Engineering*, pages 33–38, September 1987.

[27] G. Thomas, G. R. Thompson, C. Chung, E. Barkmeyer, F. Carter, M. Templeton, S. Fox, and B. Hartman. Heterogeneous Distributed Database Systems for Production Use. *ACM Computing Surveys*, 22(3):237–266, 1990.

[28] B. Thuraisingham. Multilevel Security Issues in Distributed Database Management Systems II. *Computers & Security*, 10:727–747, 1991.

[29] B. Thuraisingham and H. H. Rubinovitz. Multilevel Security Issues in Distributed Database Management Systems III. *Computers & Security*, 11:661–674, 1992.

[30] C. Wang and D. L. Spooner. Access Control in a Heterogeneous Distributed Database Management System. In *IEEE 6th Symp. on Reliability in Distributed Software and Database Systems, Williamsburg*, pages 84–92, 1987.

[31] E. Wobber, M. Abadi, M. Burrows, and B. Lampson. Authentication in the Taos Operating System. *ACM Transactions on Computer Systems*, 12(1):3–32, 1994.