An Approach For Building Secure Database Federations

Dirk Jonscher
Universität Zürich
Institut für Informatik
Winterthurerstr. 190
CH - 8057 Zürich
jonscher@ifi.unizh.ch

Klaus R. Dittrich
Universität Zürich
Institut für Informatik
Winterthurerstr. 190
CH - 8057 Zürich
dittrich@ifi.unizh.ch

Abstract

Database federations give rise to particular security problems which are not present in classical database environments. The problems and solutions heavily depend on the federation's architecture and the degree of heterogeneity of participating component systems. In this paper we discuss a special aspect of security, namely access control for tightly coupled federations. We determine the typical problems to be solved and discuss several solutions providing for different degrees of local autonomy, especially authorisation autonomy. In particular, we describe the interaction between independent reference monitors. Further, we sketch powerful access control mechanisms to be applied at the global layer and show how they can be mapped onto less powerful mechanisms of component database management systems.

1 Introduction

When building a complex information system, security requirements have to be considered right from the beginning. However, it still happens too often that systems are built by first meeting several functional requirements, and afterwards attempting to incorporate some security mechanisms.

Proceedings of the 20th VLDB Conference
Santiago, Chile, 1994

While this has not been true for relational database management systems (DBMSs) where security issues were already extensively considered for early approaches like System R (/GrWa 78/ and /Fagi 78/) and Ingres (/StWo 74/, /Ston 75/ and /StRu 76/), things look rather terrible if we consider present commercial object-oriented DBMSs. Most of them (except ORION/ITASCA) simply rely on operating system mechanisms which are at best inappropriate for DBMSs.

Database federations are still research vehicles, but there is the inherent danger that we will be faced with the same situation again. It is amazing how little work has so far been carried out to develop appropriate security mechanisms for database federations, especially if heterogeneous component systems have to be integrated, where at least one among them is not based on the relational data model.

In this paper we want to draw the attention of the database community to particular access control problems to be solved for heterogeneous database federations. Further, we discuss some approaches to tackle these problems providing for different degrees of local autonomy. These approaches are based on discretionary access control concepts with several mandatory extensions.

The contributions of this paper can be summarised as follows:

- We clarify the access control problems which arise in heterogeneous, tightly coupled federations. Note that we do not consider multilevel security concepts which we regard as an orthogonal issue (cf. /MLTS 92/, /Pern 92/, /IdQG 93/ and /LuOP 91/).
- We introduce a particular aspect of autonomy which we call *authorisation autonomy*. Further, we discuss how it is affected by the architecture of the federation.
- It is shown how different access control layers can be combined. As a highlight, we support decentralised authorisation at the global level.
- As a side-effect we have developed a novel access control concept for object-oriented DBMSs.

The remainder of this paper is organised as follows. In Section 2 we introduce the basic terminology by explaining our understanding of database federations and access control mechanisms. In Section 3 we determine the partic-

ular access control problems to be solved for tightly coupled federations. Section 4 comprises the main part of this paper, a set of access control mechanisms to be applied at the global layer of a database federation, and the mapping of these mechanisms onto less powerful ones of CDBMSs. The latter is exemplified for CDBMSs that comply with the SQL3-proposal. Related work is discussed in Section 5 and a summary is given in Section 6.

2 Basic Concepts

2.1 Database Federations

According to Sheth and Larson (/ShLa 90/), a *database federation* (federation for short) integrates existing DBMSs while preserving their autonomy. It allows for a controlled and coordinated manipulation of component database systems. Component DBMSs (CDBMSs) need not be homogeneous; they may be centralised DBMSs (having any logical data model), distributed DBMSs or even federations of their own.

From an architectural point of view, there are two classes of federations, loosely coupled systems and tightly coupled systems. Any DBMS is free to join or to leave a loosely coupled federation. It simply determines a subset of its data to be contributed to the federation (usually by means of an export schema) and it is up to users[1] of the federation to locate and access the data they are interested in. In a sense, CDBMSs act as servers. Such systems are very flexible, but the users are burdened with many responsibilities to manage the system, they have to cope with several local interfaces, etc.

Tightly coupled systems provide for global services, offered by a global DBMS, called the federated DBMS (FDBMS) that runs on top of these CDBMSs. Thus, they only need to access one interface. A federation administrator manages the federation, and global users need not worry about the location of data (location transparency), the resolution of semantic heterogeneity, and so forth. Furthermore, we assume a federation security administrator (SA) who negotiates[2] with local SAs the security policy to be applied, and sets up the global access control system. Obviously, tightly coupled systems restrict association autonomy of CDBMSs (/ShLa 90/).

Security is an issue in either case. If CDBMSs join a federation, existing applications must remain unaffected by the activities at the global layer, and integrity as well as confidentiality of data stored in the local databases have to be preserved. Especially in open systems or if the local DBMSs are administered by mutually suspicious parties (e.g. DBMSs of different companies), the federation has to provide for strong security mechanisms. Otherwise, local administrators might not agree to join the federation.

2.2 Access Control

Access rights are used to allow or forbid *subjects* (the active entities of a system, i.e. processes running on behalf of users) to execute a particular action (or operation) on a *protection object* (the assets to be protected from unauthorised accesses, e.g. files, memory segments, relations, types or classes, tuples or objects, attributes, etc.). *Access control* comprises all system *mechanisms* that are required to check whether a request issued by a particular subject is allowed or not, and all mechanisms that are required to enforce the corresponding decision. It is based on a chosen *policy* (a set of rules as authoritative regulations or directions determining what should be protected using which principles). The same mechanisms can be used to enforce different policies, and the same policy can be enforced by different mechanisms. The set of system components enforcing a particular policy determines a *security domain.*

The two main approaches to realise access control can be distinguished depending on how access rights are specified and enforced (/Denn 82/).

Discretionary access control is based on subject and protection object identities. Access rights are explicitly granted. An access right is represented as a tuple

(grantee, protectionObject, action, kindOfRight, grantor, grantOption),

indicating that the *grantee*[3] is allowed (kindOfRight='+'; *permission*) or forbidden (kindOfRight='–'; *prohibition*) to execute the *action* on the *protectionObject*. The *grantor* is the user who has granted that access right. If *grantOption* is set and the access right is a permission, the grantee is allowed to pass on that access right[4].

This kind of access control is often combined with an *ownership paradigm*, where each protection object has an owner who is a particular user responsible for granting and revoking access rights concerning this object. Since this happens at her discretion, the policy is called "discretionary". However, the same discretionary mechanisms can be combined with an *administration paradigm*, where only the security administrator is allowed to grant and revoke access rights. In this case, the policy is mandatory. In the following we will use a classification based on the underlying mechanisms.

Depending on the access rights that can be granted, several discretionary systems can be distinguished:

- positive systems: only permissions can be granted
- negative systems: only prohibitions can be granted

---

1 The same applies for applications. If we subsequently refer to "users" we mean both, users and applications. Furthermore, users of the federation are called *global users* and users of component systems *local users.*

2 Negotiation happens outside the federation. Usually, it is an ordinary conversation between humans, although it can be supported by the computer system.

3 The grantee is usually a particular user, a group of users or a role. Roles are abstract users describing the functional, organisational or social position of concrete users within the universe of discourse. Concrete users can play a role. This way, users "inherit" the rights that are granted to the corresponding role.

4 In our opinion it is not meaningful to pass on a prohibition.

- mixed systems: permissions as well as prohibitions can be granted (in this case a policy to solve conflicts is required)

Usually, the authorisation base (the set of explicitly granted access rights) is not complete, i.e. there are some requests where neither a permission nor a prohibition applies. Hence, a closure assumption is necessary:

- closed world assumption: a request is forbidden unless a permission can be inferred
- open world assumption: a request is allowed unless a prohibition can be inferred

Meaningful combinations are:

- positive/mixed system with the closed world assumption
- negative/mixed system with the open world assumption

Mandatory access control policies were developed to enforce organisation-wide security policies automatically. The most popular is *multilevel security* based on the model of Bell and LaPadula (/BeLa 75/). Multilevel security does not rely on explicitly granted access rights; instead access decisions depend on so-called security classes (or labels) that are associated with subjects (clearance levels) and protection objects (confidence levels). The main characteristic of those models is that data never flows from higher to lower security classes (unless the initiating subject is "trustworthy"). Thus, this model is inherently unidirectional. We do not describe this in more detail, since we want to focus on discretionary mechanisms. They can be combined with multilevel security (if necessary) in accordance with the TCSEC (/DoD 85/).

3 Security Problems in Database Federations

The security problems we are faced with in federations mainly stem from the heterogeneity and autonomy of CDBMSs. In the following, we consider only those problems which are typical for federations in addition to security issues which are relevant for any DBMS.

In case of loosely coupled systems, security problems are similar to those in traditional DBMSs that allow for remote accesses. Global users have to be authenticated (to verify their claimed identity), authorisation may depend on the kind of connection (remote, dial-up, local, etc.) and network security problems have to be tackled (using mutual authentication, encryption, etc.). There is no security policy for the federation as such, since there is no global authority to enforce it.

In tightly coupled systems, however, a federation authority (the federation SA) exists and the FDBMS has its own access control mechanisms. This is essential because integrated systems aggravate, for instance, privacy problems.[5] While the policy to be applied has to be agreed upon outside the computer system, database technology has to be developed providing for powerful mechanisms to enforce the chosen policy. Furthermore, independent access control within the FDBMS is necessary, because new protection objects may emerge at the global level (like global relationships, especially aggregations).

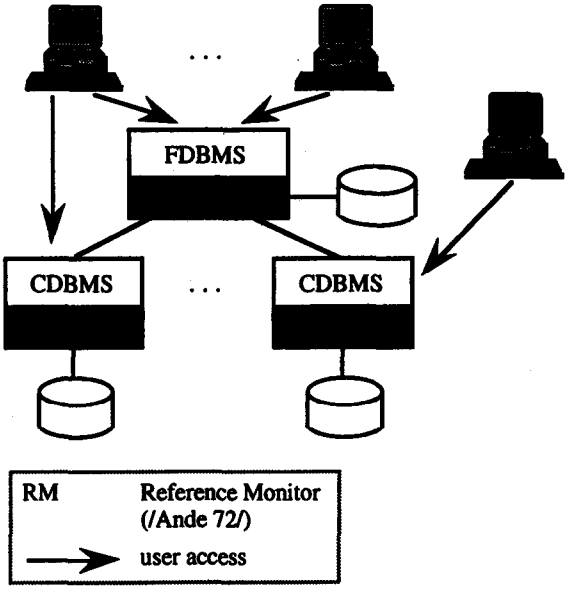


Figure 1: Two-level Access Control[6]

For the remainder of this paper, we consider tightly coupled federations only. We make the following assumptions (cf. Figure 1):

- Local access decisions take priority, i.e. a global access succeeds if and only if it is permitted by the FDBMS and the involved CDBMSs.
- The FDBMS manages its own data to store global schemas, access rights, global data, etc.
- Users can directly access CDBMSs without using the FDBMS.

Note that "security" is not a property of isolated components, but of the system in its entirety. The challenge is to establish a federation that is as least as secure as any of the CDBMSs and as transparent as possible to global users.

3.1 Heterogeneity

Besides heterogeneity with respect to hardware platforms, operating system or DBMS features of CDBMSs (like different data models, semantic heterogeneity, different query

5 Due to the integration, a larger amount of data can be accessed via a single interface, which allows for searching and combining these data. Thus, it is easier to detect global relationships between data of different CDBMSs by posing the right queries.

6 The consequences of the five-level schema architecture of Sheth and Larson (/ShLa 90/) on data security are discussed in /MLTS 92/ and /JoDi 93a/. For our purpose, this simplified view is sufficient.

languages, different transaction concepts, and so forth), we have to cope with heterogeneous local security policies and mechanisms. In the following, we focus on the latter aspect, and even further on discretionary mechanisms.

Local access control mechanisms can differ along several dimensions (see /JoDi 93a/ for a more comprehensive discussion):

- different kinds of access rights (positive, negative or mixed systems) with different closure assumptions (open vs. closed world assumption); different conflict resolution policies in case of mixed systems
- different authorisation units (users, groups or roles)
- different protection object granules (databases, domains, types/classes/relations or objects/ tuples, etc.), and different actions which can be applied on those objects
- value-independent vs. value-dependent access rights
- different authorisation paradigms (centralised vs. decentralised authorisation, ownership vs. administration paradigm); in case of decentralised authorisation, grant options vs. explicit grant permissions

Even this rather restricted scope of heterogeneity with respect to authorisation mechanisms causes considerable problems for realising access control in an FDBMS.

3.2 Autonomy

Usually, autonomy is deemed to be the key issue to distinguish database federations from distributed DBMSs. It indicates that CDBMSs retain separate and independent control over their data, even if they join a federation. Sheth and Larson (/ShLa 90/) distinguish *design, communication, execution* and *association autonomy*. We will not consider them in more detail here, but it is worth mentioning that there is a trade-off between the federation's functionality and the degree of local autonomy.

We focus on a special aspect of association autonomy which we call *authorisation autonomy*. It means that CDBMSs[7] retain control of which global users can access their data. "Authorisation" covers two different but related aspects: "to be authorised" in the sense of "to be permitted", and "to authorise" in the sense of "to confer authority upon". If the meaning is not determined by the context we use "verification" for the first aspect and "authorisation" for the second.

Different degrees of authorisation autonomy and their consequences for the federation are discussed in Section 4.

3.3 Summary

The problems to be solved can be summarised as follows:

1. provision of powerful access control concepts at the global layer to enforce elaborated security policies
2. mapping of the global concepts onto local ones by preserving authorisation autonomy (up to a certain degree)
3. protection of relationships between different CDBMSs; consideration of aggregation and inference problems which arise at the global level due to combinations of data stored in different local systems
4. information flow control if the federation is allowed to store data of component systems in its own storage areas or in other component systems
5. accountability (possibly requiring cooperation between the administrators of the federation and the local systems)
6. trustworthy authentication
7. network security

We focus on the first two issues. The others are outside the scope of this paper.[8]

4 The CHASSIS Approach

The CHASSIS[9] project aims to provide a security- and reliability-oriented integration framework to support the secure construction and operation of interoperable information systems. Its backbone is a database federation integrating heterogeneous CDBMSs (ObjectStore and Oracle were chosen for the prototype). In this section, we discuss an example where we assume a (virtual) SQL3-system as a local CDBMS. The principles, however, are more general and apply to any CDBMS (not only to relational CDBMSs like Oracle). Note that we discuss the more difficult case here, because we have to cope with different kinds of data models at the global and the local layer.

For global users, the FDBMS is characterised by its data model and security mechanisms. Therefore, we start with a brief description of both (cf. /JoMD 93/ and /JoDi 93b/ for a more comprehensive discussion). Note that neither the data model nor the security mechanisms are typical for FDBMSs, i.e. they can be applied for any DBMS. The particular aspects of federations are the authentication schemes (Section 4.4) and the combination of both access control layers (Section 4.5).

Two scenarios have to be considered (cf. Figure 1). In scenario 1, a global user issues a request to the FDBMS. The FDBMS has to verify whether this request is allowed or not. If the user passes this check, the FDBMS determines the involved CDBMSs and mediates the corresponding local requests. Two questions have to be answered:

- Which identifier (of users, roles or applications) is used by the CDBMSs for local access decisions (identification and authentication)?

---

7 Note that authorisation autonomy always refers to CDBMSs and never to the FDBMS.

8 They do not affect the mechanisms which are proposed in this paper, although some of them (4. and 5.) require additional mechanisms (3. is a design problem; discretionary mechanisms are rather a poor vehicle to solve 4. (multilevel security, e.g., seems to be more appropriate for this purpose); and the latter two concern services of the network that can be used by the FDBMS, for instance by using the Kerberos mechanisms).

9 Configurable Heterogeneous And Safe, Secure Information System (a joint project, together with the University of Geneva and ABB Baden, funded within SPP Informatik-Forschung under project number 5003-34355; cf. /NKDJ 93/)

- Is the mediated request permitted from the CDBMS's point of view (verification)?

Scenario 2 concerns decentralised authorisation at the global level. If a global user grants an access right to another global user, he needs the required permission to execute the global grant command. However, this is not sufficient to ensure that the global grantee can make use of the granted right. Authorisation autonomy requires that the involved CDBMSs agree with the global authorisation. Obviously, the second scenario is dependent on the first. Our approach aims at ensuring consistency between the global and the local authorisation states as far as possible.

4.1 Data Model

We have chosen an object-oriented data model to be applied at the global layer (/JoDi 93b/). The two main reasons for this decision are as follows:

- We have to cope with heterogeneous CDBMSs, probably including object-oriented ones. Thus, we need a powerful data model subsuming the concepts of local systems.
- Object-oriented data models are a well-accepted vehicle to establish heterogeneous federations, especially because of their support for encapsulation and polymorphism which allow for hiding heterogeneity (/SaCG 91/, HäDi 92/, /NiWM 93/ and /Kent 93/).

For the purpose of this paper, there is no need to describe this model in more detail. It offers the usual features like object identity, encapsulation, types and subtyping, (multiple) inheritance and complex objects (/Atki 89/).

4.2 Access Control Mechanisms

In this section we describe the mechanisms provided at both access control levels of the federation. They form the basis for Section 4.5.

4.2.1 Mechanisms of the FDBMS

The main features of the security mechanisms at the global level can be summarised as follows:

- CHASSIS provides for a mixed system with the closed world assumption. In case of conflicts, prohibitions override permissions.
- Access rights can be granted to individual users and to roles.
- Users can be associated with multiple roles. Association means that these roles can be activated by the user (using a particular command). A user is allowed to activate multiple roles he is associated with, even at the same time. Restrictions to prevent forbidden accumulations of rights and to support separation of duties principles (/ClWi 89/) can be described by conflict relations. An *association conflict relation* describes which roles a user can never be simultaneously associated with (separation of duties). The *activation conflict relation* prescribes which roles can never be simultaneously activated.
- Roles can be related by subordination relationships (a reflexive, antisymmetric and transitive binary relationship between roles). These relationships are used to infer implicit rights. A role "inherits" permissions from its subordinated roles and prohibitions from its superior roles. This results in a system where superior roles have strictly greater power (or authority) than their subordinates.
- Subjects as well as protection objects (which coincide with (complex) objects of the data model) can be grouped by means of *subject* or *protection object domains*, respectively. Domains of the same kind can be nested. Access rights (permissions or prohibitions) which concern a domain apply to any of its elements.
- Access rights allow or forbid the execution of methods.[10]
- Several rules exist to infer implicit rights according to the data model: access rights for complex objects apply to any of its component objects, too; the permission to define subtypes of a given type implies the permission to read its description, and the permission to execute a method implies the permission to read its signature.
- Authorisation is based on the following principles:
  - Every object has an owner who is a particular user. A predefined owner *system* is used for objects where an administration paradigm should be applied.
  - Users are by default not allowed to participate in decentralised authorisation. Grantors need the privilege allowing for granting (*giveAuthority*-privilege) and grantees the privilege allowing for receiving (*getAuthority*-privilege) access rights. These privileges are given to roles and apply to associated users. SAs are permitted to override these privileges (they can grant/get rights to/from any user; note that *security administrator* is a special role, and that these privileges are only necessary but not sufficient to grant a concrete right).
  - Owners can grant or revoke any access right (including authorisation rights) to/from any grantee (supposing both sides have the required privileges to participate in decentralised authorisation). In case of system-owned objects, the SA acts on behalf of the owner *system*.
  - Decentralised authorisation can be based on grant flags (as usual for relational DBMSs) or on explicit grant permissions.
  - Every user can revoke rights that he has granted. In case of authorisations based on grant flags, a recursive revocation scheme is applied (according to /GrWa 78/ and /Fagi 78/).

---

10 We consider this as being essential for access control concepts which are based on object-oriented data models. Authorisation has to be consistent with encapsulation. Note that this is different from the ORION scheme (/RaWK 88/ and /RBKW 91/), although Bertino (/Bert 92/) has suggested a method-based extension.

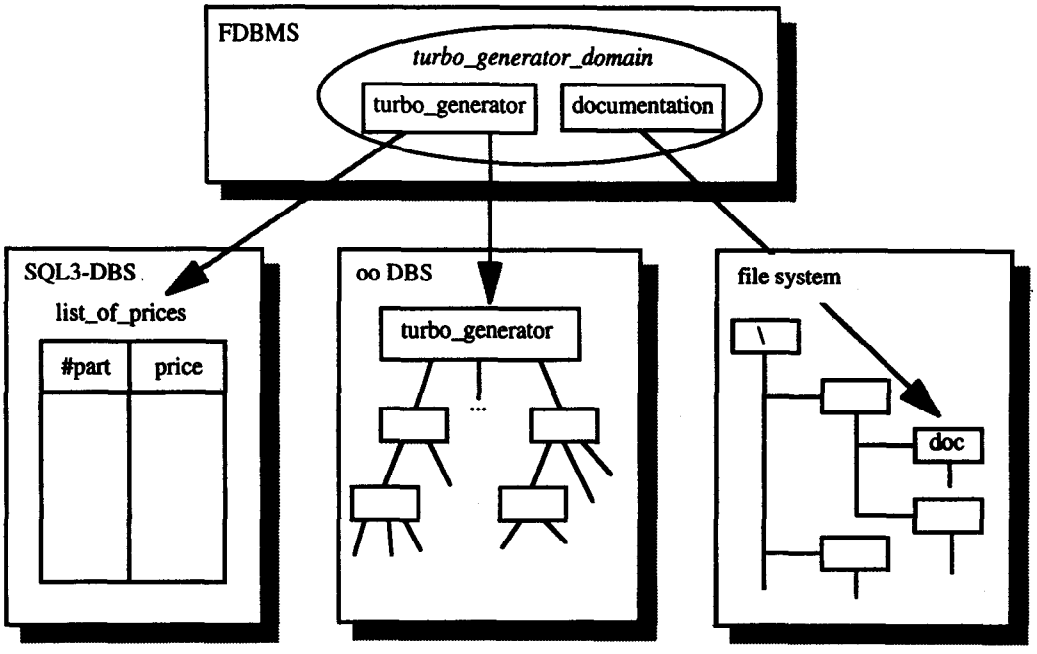


Figure 2: Example Application

These mechanisms are powerful enough to implement several important security principles like "separation of duties" (/ClWi 87/), authoriser roles, hierarchical security administrators, etc. It is possible to decide for every object whether it should be administered by a central authority (administration paradigm) or in a decentralised manner (ownership paradigm). Even if an ownership paradigm is applied, the security administrator retains control who is allowed to grant access rights to whom (see /JoDi 93b/ for a more comprehensive discussion).

4.2.2 Mechanisms of the CDBMS (SQL3-DBMS)

The following mechanisms are suggested by the SQL3-proposal (/Melt 93/):

- a positive system with the closed world assumption
- users and roles as authorisation units, where roles are placeholders for sets of permissions which can be completely granted to users or other roles (i.e. roles can be nested); users can be associated with several roles, but roles have to be activated (enabled) explicitly; activating a role implies deactivating the previously activated one
- decentralised authorisation is based on an ownership paradigm and on grant flags (*administration options* in case of roles, *grant options* in case of access rights)

4.3 Running Example

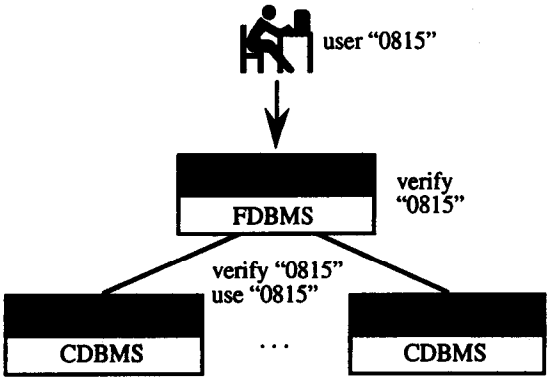
To clarify the concepts, we use an example as sketched in Figure 2, where we focus on the combination of the FDBMS and the SQL3-DBMS. We assume a design environment where a team is developing turbo-generators. The team consists of a (project) *manager* and two subordinated roles, *designer* and *programmer*. The data which are required for this project are kept by an FDBMS defining a protection object domain *turbo_generator_domain*.[11] It consists of a (complex) type *turbo_generator* (concrete turbo-generators are instances of this type) and its documentation. The documentation is simply kept by a file system (a set of text files stored in a dedicated subdirectory). Federated turbo-generators (type *turbo_generator*) are composed of concrete generators which are stored as complex objects in an object-oriented CDBMS and some auxiliary information which is kept by an SQL3-system. The latter provides for information like prices of parts, etc. that can be used, e.g., to calculate the price of a particular generator at the global layer. Subsequently, we only refer to a method *price()* (of the global type *turbo_generator*) which has to access the corresponding generator object to count which parts are how often used (local methods *parts()* and *count_part()*), and has to access the relation *list_of_prices* of the SQL3-DBMS to get the actual prices of parts (local *Select*-operator).

4.4 Authentication Schemes

Authentication is the process of verifying the claimed identity of a user. In the following, we do not discuss *how* this identity can be verified (using passwords or whatsoever), but focus on *who* validates that identity (the answer is related to the first question of scenario 1). We

[11] Thus, powerful permissions like "(manager, turbo_generator_domain, ALL, +, userXYZ, 1)" can be granted.

discuss two different schemes: local (Figure 3) and global authentication (Figure 4). The latter is further subdivided into global authentication with and without delivering the identity of global users.[12] Authorisation autonomy is decreasing whereas the federation's functionality is increasing from I to IIb. The schemes determine which identity is used for local access control decisions. Thus, they are related to the second question of scenario 1. Obviously, they also affect scenario 2.

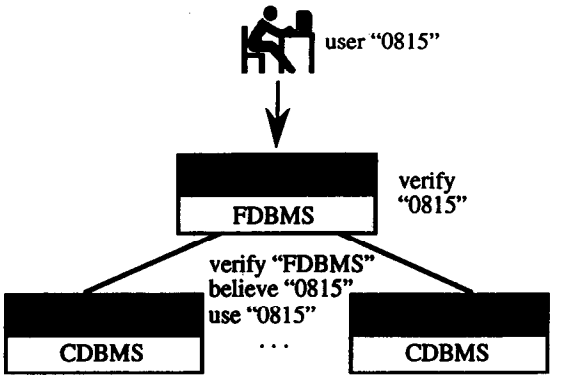


I) full authorisation autonomy

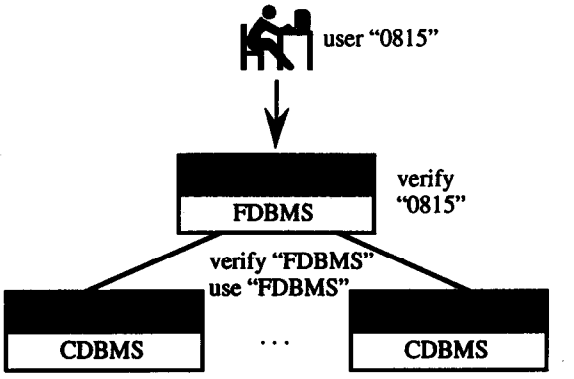
Figure 3: Local Authentication

Full authorisation autonomy can be preserved if the CDBMSs again validate the identity of global users.[13] Local access decisions are based on local user identifiers. There is no need to validate that the request was mediated by the FDBMS (architecture I). This approach is rather cumbersome for global users, since they have to type in multiple passwords.

In the second case (medium authorisation autonomy), the CDBMSs put some trust into the FDBMS that it correctly validates the identity of global users.[14] They simply require that the FDBMS authenticates itself and delivers the identity of the requesting user (the identity which is known to the local system) together with the corresponding subrequest. This identifier is used for local access decisions. In a sense, the FDBMS and the CDBMSs form a unit of authentication from the global users' point of view. The difference to scheme I is that the component systems have to trust the FDBMS not to deliver wrong identifiers.



IIa) medium authorisation autonomy



IIb) low authorisation autonomy

Figure 4: Global Authentication

In the third scheme (low authorisation autonomy), only the FDBMS validates the identity of global users whereas local systems do not worry about it. They simply require that the FDBMS authenticates itself, and use its identity for local access decisions. Thus, the local rights which are given to the FDBMS implicitly determine the "export schema" of the CDBMS. The corresponding data are freely available to the FDBMS and further restrictions are only enforced at the global layer. This is very convenient for global users, but requires a considerable amount of trust into the security mechanisms of the FDBMS.

Note that different CDBMSs can choose different schemes within the same federation. Therefore, the FDBMS has to keep track which scheme was chosen by a concrete CDBMS and has to apply the proper protocol as discussed in the next section.

4.5 Combining both access control levels

In order to combine both access control levels and preserve authorisation autonomy, we need to map the global access rights onto local ones. This does not mean, however, that

12 Such a classification emphasises the authentication aspect. Another possibility is to stress the identification aspect which allows for grouping the schemes I and IIa. The latter classification is related to the verification procedure.

13 For simplicity, we assume that user identifiers are unique within the whole system. However, it is very likely that a global user has different identifiers for the federation and for every CDBMS. Hence, the FDBMS has to keep track of the corresponding mappings if a global authentication scheme is applied.

14 An alternative to trusting the FDBMS with respect to authentication is to involve a global authority, e.g. a dedicated authentication server.

the local rights are automatically generated, because CDBMSs may refuse to grant a requested local right (depending on their local security policy).

Another problem is caused by the object-oriented data model of the FDBMS. The global access control component can hardly know which local objects are how involved in the execution of a global method; this information is buried within the methods' code. Hence, the FDBMS needs support to map global access rights onto local ones. What we need is a relationship between global and local *capabilities* (pairs of protection objects and actions). At the global layer, we are faced with capabilities to execute methods. The designer of the method has to declare which local capabilities are required to execute this method successfully.

For our example method *price()*, we have the following relationships (identifiers of the corresponding DBMS are included to resolve name ambiguities):

(FDBMS, turbo_generator, price) :
(SQL3-DBMS, list_of_prices, SELECT)
(oo DBMS, turbo_generator, { parts, count_part })

The mechanisms provided at the global layer (cf. Section 4.2.1) are powerful enough to subsume the usual discretionary mechanism of local systems, but not vice versa. Hence, additional concepts are required to ensure a consistent authorisation state between the FDBMS and the CDBMS (as far as possible). The following principles are applied:

- Access rights which are based on set expressions (roles, subject domains, protection object domains) are mapped onto a set of local rights for every element of the corresponding set if the CDBMS does not support an equivalent of the global concept.
- Grant permissions are mapped onto applicable access permissions with grant option.
- The FDBMS transparently activates the required local role such that a local subrequest will succeed.

Another question to be addressed is how the FDBMS should react if local systems refuse to grant the required local rights which correspond to a global grant (scenario 2). It is our belief that a global right is only meaningful if all corresponding local rights can be granted or are already available for the grantee (possibly granted by a different grantor). It is thus preferable to reject the global grant as long as the corresponding local rights are not available. However, the global grantor needs some explanation *why* the grant failed[15] to support the direct negotiation with local authorities.

An analogous problem occurs if a CDBMS revokes a local access right that is required to execute a global method successfully. Usually, CDBMSs will not notify any change of their authorisation state to the FDBMS. Thus, the FDBMS becomes aware of this problem if an attempt is made to execute the affected global method. According to our scheme for global authorisations, the global access right is automatically revoked if it does no longer comply with the local authorisation state (a kind of "lazy propagation"). Furthermore the grantor of the revoked access right is notified.

First we consider the most difficult case where the CDBMS wants to retain full authorisation autonomy. We describe how to cope with global concepts which are not supported by a local SQL3-DBMS. Although exactly the same problems arise for medium authorisation autonomy, the latter scheme allows for a trick to work around some restrictions of the former scheme (this trick, however, decreases authorisation autonomy). All the problems disappear if we apply a scheme where CDBMSs are satisfied with low authorisation autonomy.

4.5.1 Full authorisation autonomy

If full authorisation autonomy has to be preserved, the FDBMS has to ensure that its authorisation state is compatible with the authorisation state of all involved CDBMSs. However, an SQL3-DBMS does not support prohibitions, roles to be activated concurrently, role conflict relations, subject or protection object domains and elaborated authorisation mechanisms. Therefore, we define the following mappings:

Prohibitions

Two approaches are possible:

1. Ignore prohibitions and ask for local permissions for every global permission.
2. Try to get local permissions for those global permissions which are not overridden by prohibitions.

The former approach is very easy but causes an amplification of rights if global users can also directly access the CDBMS. The latter approach, on the other hand, is much more difficult than it seems to be. In this case, the FDBMS cannot benefit, for instance, from the local role concept. At the global layer, a permission for *designer* and a prohibition for a particular designer (say "Bob") may exist. Hence, the FDBMS has to determine who is currently associated with this role, and must ask for a local permission for every single developer, except Bob. It also has to keep track of changes concerning the association between users and roles. Although this is possible, it causes a considerable overhead if the FDBMS is running in a dynamic environment (the price to be paid for authorisation autonomy).

Furthermore, authorisation at the global level is not simply based on ownership or grant flags. Since also global grant permissions can be granted, it is likely to happen that a global user wants to grant a prohibition. Granting a global prohibition, however, requires that the grantor is allowed to revoke conflicting local permissions of the grantee. Obviously, such a constellation is purely accidental. Therefore, such an authorisation will usually fail.

---

15 If the grantor is not the designer of the method, he cannot know which local access rights are required. Therefore, the FDBMS has to deliver the list of local capabilities that were rejected (supposing the grant was allowed from the federation's point of view).

We argue for the first approach which was chosen for CHASSIS. An amplification of rights can be avoided if it is feasible to prevent global users from accessing CDBMSs directly.[16]

Roles

Roles at the global layer can be mapped onto local roles. Assume that *designer* should be allowed to execute the method *price()*. This implies an analogous permission for managers and requires a local *Select*-permission for the relation "list_of_prices". The FDBMS can make use of the local authorisation interface (we assume that the global and the local role names are identical, but the FDBMS can also keep track of mappings between different names; furthermore, we assume that the local roles already exist):

GRANT SELECT ON list_of_prices TO designer
GRANT designer TO manager

If the relationship between the local roles has already been established, we can omit the second command. If the global grantor is not allowed to grant the corresponding local access rights, the global grant is also rejected.

In the general case, it may happen that the local system does not agree with the grantees[17]. Then we can try some work-arounds:

GRANT SELECT ON list_of_prices TO manager
or: GRANT SELECT ON list_of_prices
TO <user_list>

(where <user_list> comprises all users belonging to *manager* or *designer*)

If a particular user issues an authorised request to execute the method *price()*, the FDBMS must determine the role for which the request is allowed and has to activate the corresponding local role. Thus, it is not sufficient to mediate simply the query to the CDBMS. At first, a SET ROLE command may be necessary:

SET ROLE designer/manager
SELECT #part, price FROM list_of_prices
WHERE ...

This way, a global user can activate several roles at the global layer and needs not worry about switching between local roles.

The activation conflict relation is not a problem for an SQL3-DBMS, because it is impossible to activate roles concurrently. The association conflict relation, however, is not supported.

Domains

Domains are a simple mechanism to group sets of subjects (users, roles or other subject domains) or protection objects. The FDBMS simply determines their basic elements (by flattening the domain structure) and attempts to grant the corresponding rights individually.

Authorisation

Authorisation is the most complicated issue for a scheme where full authorisation autonomy has to be preserved. The FDBMS can only give some support to determine which local rights are required, and it can try to get them if the CDBMS provides for an authorisation interface (grant and revoke commands).

A global authorisation can only succeed if the global grantor is either the owner of affected local objects, or possesses the local access right to be granted with grant option. If a global permission is given with grant option, the corresponding local grant commands are also generated with grant option (if the local grant can succeed it will also succeed with grant option), e.g.:

GRANT PERM price ON turbo_generator TO Jim
WITH GRANT OPTION[18]

is translated into the corresponding local right:

GRANT SELECT ON list_of_prices TO Jim
WITH GRANT OPTION

The matter becomes more complicated if we want to grant a grant permission at the global layer. Such a permission allows for granting *any* access right for the corresponding object at the global layer. Hence, we need to consider any method that can be executed for this object and have to apply for the corresponding local access permissions with grant option (if the grantee is not the owner of the local object). In the general case, this will fail. However, the system can acknowledge which local requests were successful and which ones were not. Afterwards, the global grantor can try to persuade the local authorities to get the required local access rights.

Restrictions like who is allowed to participate in decentralised authorisation (getAuthority- and giveAuthority-privileges) have no local equivalent. Furthermore, there is no equivalent for the global administration paradigm. The best local approximation is to grant the corresponding local access rights to federation SAs (as a role).

4.5.2 Medium authorisation autonomy

In the previous section, we have shown that decentralised authorisation at the global level is severely restricted if full authorisation autonomy has to be preserved. Although the FDBMS offers mechanisms that are much more powerful than any of the DBMSs commercially available today, the CDBMSs cannot much benefit from them.

However, if the CDBMSs do not require that global users have to authenticate themselves locally, the FDBMS can use a technique[19] that we call *subject switching*.

---

16 There are several solutions to achieve this restriction. For instance, such users may not be able to start a corresponding client process, because they are not allowed to execute the corresponding program.

17 Such a restriction is not possible in case of an SQL3-system; either any grant is allowed (based on grant options or the ownership position), or no grant at all.

18 Depending on what kind of right is to be granted, we suggest different key words: GRANT PERM (permission) and GRANT PROH (prohibition).

19 It has to be negotiated between local and global authorities whether this technique is allowed or not.

Every access right at the global layer has an attribute determining the grantor of this right. Assume that a user, say Betty, who has the permission to execute the local query, wants to grant the permission to execute the method *price()* to Bill, and Betty is allowed to grant this permission (from the federation's point of view). In this case, the FDBMS can generate a global permission for Bill without applying for any local access right. If Bill wants to execute the method *price()*, the FDBMS allows the request and generates the query not on behalf of Bill, but on behalf of Betty. Since Betty has the required local permission, the request succeeds. This approach requires some cooperation between local and global authorities to ensure accountability (the local system assumes that Betty has issued the query, and only the FDBMS can testify that it has not been Betty, but Bill).

An unrestricted subject switching is not meaningful. In this case it would be better to choose low authorisation autonomy for the CDBMS. Instead it should be negotiated under which circumstances the FDBMS is allowed to apply this technique (e.g., only for particular local objects or only for particular users). Such a restriction has to be a part of a contract, in particular if both parties do not belong to the same security domain (i.e. if both, the FDBMS and the CDBMS do not belong to the same company). The local authorities need a reliable way to access the relevant part of the global audit trail in order to verify whether the global system complies with the contract or not.

4.5.3 Low authorisation autonomy

This is the most convenient case for the federation as well as for global users. The FDBMS simply has to care for getting the required local permissions for itself (as a particular local "user") and decentralised authorisation, etc. at the global layer is completely transparent for the CDBMSs. In a sense, the FDBMS serves as a secure front-end for the CDBMSs.

Using this scheme, the local systems can benefit the most from the federation's security mechanisms. It is even possible to protect CDBMSs which do not provide for elaborated access control facilities by integrating them into the FDBMS. It "simply" has to be ensured that untrusted users cannot directly access the CDBMS.

Since security is a property of the system in its entirety, we can imagine many applications where it is meaningful to sacrifice authorisation autonomy to increase the system's security.

Summing up, scheme I realises two different security domains, whereas scheme IIb implements a nested security domain. Scheme IIa is a hybrid one with two overlapping security domains.

5 Related Work

The Mermaid mechanisms (/TeLW 87/) have represented the state of the art of access control for federations for a surprisingly long time. Mermaid is a front-end system to integrate multiple homogeneous relational DBMSs by ensuring distribution transparency. Authorisation autonomy has been preserved. Access rights are individually granted at the global level, but do not imply any right for involved CDBMSs (i.e. the corresponding local rights have to be granted explicitly). Access control is based on access control lists which are associated with certain schemas. A user having an entry within such an access control list is allowed to carry out the corresponding relational operator for any relation that belongs to this schema. Local access validations are carried out independently. A mechanism was proposed to simplify remote logins of global users (Mermaid is allowed to store the corresponding local identifiers and passwords of global users to establish remote logins automatically). Thus, Mermaid provides for a two-level access control and full as well as medium authorisation autonomy. However, it only integrates relational CDBMSs and does not support decentralised authorisation at the global layer.

Wang and Spooner (/WaSp 87/) describe an access control mechanism for heterogeneous federations (supporting relational and network CDBMSs) which is based on views and an ownership paradigm. Their approach to achieve authorisation autonomy is to provide only snapshots of local data to global users. Thus, global write accesses are not supported.

Sheth and Larson (/ShLa 90/) have discussed some access control problems based on their five-level schema architecture, but not in much detail. Protection is simply based on views.

A more comprehensive discussion is given in /MLTS 92/, although the focus is rather on multilevel security. The parts concerning discretionary mechanisms are also based on views.

Pernul (/Pern 92/) discussed a security model for a homogeneous tightly coupled federation aimed at integrating autonomous relational DBMSs with different local security mechanisms. Local systems can apply a discretionary policy, a mandatory policy or a combination of both. The global canonical model comprises the functionality of local systems and allows for defining additional access restrictions at the global level. However, the focus of the author was on mandatory access control mechanisms, whereas the discretionary ones are close to Mermaid.

Another approach based on mandatory mechanisms can be found in /IdQG 93/.

6 Summary and Outlook

In this paper, we have shown how access control based on discretionary mechanisms can be achieved for tightly coupled database federations.

We have chosen an object-oriented data model to be applied at the global layer to cope with heterogeneous CDBMSs. The federation's access control mechanisms are very powerful and provide for role and domain concepts and an elaborated decentralised authorisation (including several mandatory extensions of the usual ownership paradigm).

Furthermore, we have shown how different authentication schemes within the federation have an impact on au-

thorisation autonomy. We have explained the trade-off between authorisation autonomy and the functionality of the FDBMS as well as the security of the whole information system.

Finally, we discussed a combination of both access control levels exemplified by an SQL3-CDBMS.

Two additional features of our security mechanisms have not been mentioned in this paper:

- *Predicates* can be used to restrict access rights (cf. /FeSW 81/ and /JoDi 93b/). They are the key to support several different security policies using the same mechanisms. In principle, they are able to enforce policies being as mandatory as necessary. In particular, they can be used to restrict which access rights can be granted to whom.
- So-called *complex subjects* support more elaborated access control policies which involve multiple users (e.g. n-person rules).

Currently, we are implementing a CHASSIS prototype using a C++ environment where a database federation is built on top of ObjectStore and Oracle7. ObjectStore is also used to keep the required schema information of the FDBMS.

Acknowledgement

We would like to thank Jonathan Moffett from the University of York for many beneficial discussions and criticisms. In particular, he has pointed out the ambiguity of the notion "authorisation", and the different possibilities to classify the architectures providing for different degrees of authorisation autonomy depending on whether the identification or the authentication aspect is emphasised.

References

/Ande 72/ Anderson, J.P.; *Computer Security Technology Planning Study*; ESO-TR-73-51 (AD-758206), J.P. Anderson Co., Oct. 1972

/Atki 89/ Atkinson, M.; Bancilhon, F.; DeWitt, D.; Dittrich, K.; Maier, D.; Zdonik, S.; *The Object-Oriented Database System Manifesto*; 1st International Conference on Deductive and Object-Oriented Databases, Kyoto, Dec. 1989

/BeLa 75/ Bell, D.E.; LaPadula, L.J.; *Secure Computer Systems: Unified Exposition and Multics Interpretation*; MTR-2997, Mitre, Bedford, MA, 1975

/Bert 92/ Bertino, E.; *Data Hiding and Security in Object-Oriented Databases*; Proc. 1992 Int. Conf. on Data Engineering, IEEE Computer Society Press, Phoenix, Feb. 1992, 338-347

/ClWi 87/ Clark, D.D.; Wilson, D.R.; *A Comparison of Commercial and Military Computer Security Policies*; Proc. IEEE Symp. on Security and Privacy, Oakland, Apr. 1987, 184-194

/Denn 82/ Denning, D.E.; *Cryptography and Data Security*; Addison Wesley, MA, 1982

/DoD 85/ Department of Defense; *Trusted Computer System Evaluation Criteria*; DOD 5200.28-STD, Department of Defense, Dec. 1985

/Fagi 78/ Fagin, R.; *On an Authorisation Mechanism*; ACM Transactions on Database Systems, Vol. 3, No. 3, Sep. 1978, 310-319

/FeSW 81/ Fernandez, E.B.; Summer, R.C.; Wood, Ch.; *Database Security and Integrity*; Addison-Wesley, MA, 1981

/GrWa 76/ Griffith, P.P.; Wade, B.W.; *An Authorization Mechanism for a Relational Database System*; ACM Transactions on Database Systems, Vol. 1, No. 3, Sep. 1976, 242-255

/HäDi 92/ Härtig, M.; Dittrich, K.R.; *An Object-Oriented Integration Framework for Building Heterogeneous Database Systems*; Proc. of the IFIP DS-5 Conference on Semantics of Interoperable Database Systems, Lorne, Australia, Nov. 1992

/IdQG 93/ Idris, N.B.; Qutaishat, M.A.; Gray, W.A.; *Integration of Secrecy Features in a Federated Database Environment*; Proc. IFIP WG 11.3 7th Annual Working Conference on DB Security, Huntsville, AL, Sep. 1993, 84-109

/JoDi 93a/ Jonscher, D.; Dittrich, K.R.; *Access Control for Database Federations; a discussion of the state-of-the-art*; Proc. DBTA Workshop on Interoperability of DBSs and DB Applications, Fribourg, Switzerland, Oct. 1993, 156-178

/JoDi 93b/ Jonscher, D.; Dittrich, K.R.; *A Formal Security Model based on an Object-Oriented Data Model*; Technical Report No. 93.41, Institut für Informatik der Universität Zürich, Nov. 1993

/JoMD 93/ Jonscher, D.; Moffett, J.D.; Dittrich, K.R.; *Complex Subjects; or: The Striving for Complexity is Ruling our World*; Proc. IFIP WG 11.3 7th Annual Working Conference on Database Security, Huntsville, AL, Sep. 1993, 18-36

/Kent 93/ Kent, W.; *Object Orientation and Interoperability*; NATO Advanced Study Institute, Kusadasi, Turkey, Aug. 1993

/LuOP 91/ Lu, H.; Ooi, B.-C.; Pang, H.H.; *Multilevel Security Control in Multidatabase Systems*; Proc. 1st Int. Workshop on Interoper-

ability in Multi-DBSs, Kyoto, Japan, Apr. 1991

/Melt 93/ Melton, J. (ed.); *Database Language SQL (SQL3)*; ISO-ANSI Working Draft, Aug. 1993

/MLTS 92/ Morgenstern, M.; Lunt, T.F.; Thuraisingham, B.; Spooner, D.L.; *Security issues in federated database systems: panel contributions*; Landwehr, C.E.; Jajodia, S. (eds.), Database Security, V: Status and Prospects, IFIP, Elsevier, 1992, 131-148

/NiWM 93/ Nicol, J.R.; Wilkes, C.Th.; Manola, F.A.; *Object Orientation in Heterogeneous Distributed Computing Systems*; IEEE Computer, Jun. 1993, 57-67

/NKDJ 93/ Nierstrasz, O.; Konstantas, D.; Dittrich, K.; Jonscher, D.; *CHASSIS: A Platform for Constructing Open Information Systems*; Proc. AFCET '93, Versailles, Jun. 1993, 153-161 (French version);

also published in: Tsichritzis, D. (ed.); Visual Objects; Université de Genève, Centre Universitaire d'Informatique, Jun. 1993, 235-245 (English version)

/Pern 92/ Pernul, G.; *Canonical Security Modeling for Federated Databases*; Proc. of the IFIP DS-5 Conf. on Semantics of Interoperable DBSs; Lorne, Australia, Nov. 1992, 199-214

/RaWK 88/ Rabitti, F.; Woelk, D.; Kim, W.; *A Model of Authorization for Object-Oriented and Semantic Databases*; Proc. of the Int. Conf. on Extending Database Technology, Venice, Italy, Mar. 1988, 231-250

/RBKW 91/ Rabitti, F.; Bertino, E.; Kim, W.; Woelk, D.; *A Model of Authorization for Next-Generation Database Systems*; ACM Transactions on Database Systems, Vol. 19, No. 1, Mar. 1991, 88-131

/SaCG 91/ Saltor, F.; Castellanos, M.; Garcia-Solaco, M.; *Suitability of data models as canonical models for federated databases*; SIGMOD Record, Vol. 20, No. 4, Dec. 1991

/ShLa 90/ Sheth, A.P.; Larson, J.A.; *Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases*; ACM Computing Surveys, Vol. 22, No. 3, Sep. 1990, 180-236

/Ston 75/ Stonebraker, M.; *Implementation of Integrity Constraints and Views by Query Modification*; Proc. ACM SIGMOD Conference on Management of Data, San Jose, CA, May 1975, 412-418

/StRu 76/ Stonebraker, M.; Rubinstein, P.; *The Ingres Protection System*; Proc. ACM Annual Conference, 1976

/StWo 74/ Stonebraker, M.; Wong, E.; *Access Control in a Relational Data Base Management System by Query Modification*; Proc. ACM National Conference, San Diego, CA, Nov. 1974

/TeLW 87/ Templeton, M.; Lund, E.; Ward, P.; *Pragmatics of Access Control in Mermaid*; Data Engineering, Vol. 10, No. 3, Sep. 1987 (Special Issue on Federated Database Systems), 33-38

/WaSp 87/ Wang, C.-Y.; Spooner, D.L.; *Access Control in a Heterogeneous Distributed Database Management System*; IEEE 6th Symp. on Reliability in Distributed Software and Database Systems, Williamsburg, VA, Mar. 1987, 84-92