On Designing a Database for Integrated User Management: Pitfalls and Possibilities

Amy LaMeyer

Shankaranarayanan Ganesan

Jesper M. Johansson

Boston University

Abstract

Decisions on implementing IT systems have often been departmental or isolated in nature. As a result many organizations now are faced with the challenge of integrating different networks and computers (in different departments or possibly even within each), each managed by a different operating system, and each running different types of applications. In the last decade all organizations have understood the importance of integrating the different systems and applications. While practitioners and researchers have proposed and implemented several methods for integrating systems, applications, and data, one area has been and continues to be difficult to integrate - user accounts.

In this paper we present an approach to integrate user account information from several systems. This approach is interesting for several reasons. First it is not specific to managing users on a particular system and is general enough to be used in an integrated environment that includes several systems of different types. Second, it is easily scalable to include applications and networks that have users and need user management. Third, it is simple and easy to understand and implement. The approach results in a database of user accounts that can be queried for specific users, groups, and so on. The design can be implemented in any relational database, or for example in the Windows 2000 Active Directory to take advantage of the fact that it already tracks user accounts and related information.

1. Introduction

Decisions on implementing IT systems have often been departmental or isolated in nature. As a result many organizations now are faced with the challenge of integrating different networks and computers (in different departments or possibly even within each), each managed by a different operating system, and each running different types of applications. In the last decade all organizations have understood the importance of integrating the different systems and applications. While practitioners and researchers have proposed and implemented several methods for integrating systems, applications, and data, one area has been and continues to be difficult to integrate - user accounts.

There may be several reasons why user account integration has not received a great deal of attention. Every enterprise system supports the management of users on that system. As a matter of fact, it is a basic security requirement (C1) that the OS be capable of identifying users (Department of Defense 1985). To further complicate matters, many applications, such as databases, define their own user accounts. Many systems have sophisticated user account databases that can store information not just about the user accounts, but comments about the users as well. Windows NT 4.0, for example, supports this, and the schema used to store user accounts in Windows 2000 can be extended to support a much richer set of information about each user. On the other hand, it is not at all uncommon to find systems that do not even support storing a user's real name, not to mention critical identifying information such as employee ID. This invariably results in an extensive duplication of user information, and the inability to answer specific question such as what access privileges a specific user of a system has on other systems in the organization, or even fundamental questions

such as which employees have access rights on a specific system. Employees may also have different user account names on different systems. Organizations have often chosen not to deal with this problem, as they understand it to be vicious cycle. If the organization could track all of its IT users across all systems user management would be considerably simpler. To resolve this issue many organizations see the need to manage users in an integrated fashion, which is difficult, if not impossible, given the obscure status of current user information. The problem has therefore not been addressed and no feasible, practical solution suggested.

In this paper we present an approach to integrate user account information from several systems. This approach is interesting for several reasons. First, it is not specific to managing users on a particular system and is general enough to be used in an integrated environment that includes several systems of different types. Second, it is easily scalable to include applications and networks that have users and need user management. Third, it is simple and easy to understand and implement. The approach results in a database of user accounts that can be queried for specific users, groups, and so on. The design can be implemented in any relational database, or, for example, in the Windows 2000 Active Directory to take advantage of the fact that it already tracks user accounts and related information.

1.1 User Accounts

Each system (computers or networks based on NT, UNIX, VMS, etc.) supports ways of managing users on that system. Microsoft's Active Directory is designed to support the management of users enterprise-wide in a Windows-2000 server-based network, for example. Specifically, Active Directory addresses the management of users in terms of the privileges and resources available to or accessible by each. Other software firms have created and made available software to comprehensively manage users in a specific type/network of computers such as UNIX (e.g. COSadmin by Open Systems Management). It is important to note that all of the above address user management in a single, specific system or network. There are approaches that present exceptions, such as Novell's Netware Directory Services (NDS), which does work both on Netware and Windows NT/2000. However, it is still limited to managing user accounts on those systems for which it was designed. Our approach does not substitute these user management features but supplements them and helps pull them together by providing a design for a central location to track user accounts.

1.2 Identifying User Accounts

To completely understand the issues involved, let us briefly visit these. Individual computer systems typically have users identified either directly by a username (as in UNIX based implementations) or by a unique user identifier (the SID on NT based systems). Individual systems may be part of one or more computer network(s) and these have users who need to be managed. Further, large shared applications such as an application database, email, knowledge management subsystems etc. may use the user account database of the host OS, or support their own accounts. Some databases, such as Microsoft's SQL Server, support both approaches. Very often, however, the access to such applications is independent of the access to the computer systems they run on. Most of the above systems manage privileges assigned to users using groups or roles. We now have different systems that users have access to, different groups within each system that define and manage user privileges, different applications within each system that are accessed by users, and networks/sub-networks that connect these systems together. The proposed design for managing users in an integrated fashion addresses a majority of the above issues and can be extended to deal with the rest as well. We do not deal with privileges and access to specific resources in this paper, as individual systems provide good support for managing privileges and resources within each. As a first step, we address the other user management issues that are not adequately supported.

The remainder of the paper is organized as follows. The specific problem that motivated this paper is described in detail in Section 2. For convenience, and to protect the identity of the organization, we henceforth refer to the organization as Information Systems Inc. (ISI. Also described in Section 2 are the existing systems and corresponding user accounts, and the need for integrating the user management as well as specific issues that need to be addressed in this process. Section 3 presents a design of a database that incorporates existing accounts and integrates the management of users in the organization. A discussion of how to utilize some of the features of Windows 2000 to make the user identification task easier, are presented in Section 4 and the issues that need to be addressed to resolve problems in importing the existing data into this schema are described here. Concluding remarks, directions for improvement, as well as directions for further work are presented in Section 5.

2. The Current State of the Systems and User Accounts at ISI

ISI Inc. is an organization that provides a variety of high quality health care solutions. These are directed towards individual customers as well as for use in hospitals and home health care services. The categories of products include pharmaceutical, nutritional, diagnostic, and hospital supplies. ISI Inc. owned and operated a subsidiary organization that manufactured, sold, and more importantly provided customer service and support for some of the products manufactured at ISI that required customer service and post-sales support. ISI Inc., like any other organization today, uses IT and IT systems to store and manage their extensive data resources as well as maintain the applications that capture and make use of this data. Typical applications include the standard corporate applications for sales force planning, sales and demand forecasting, financial applications, human resource management, customer service and support, as well as applications for monitoring and planning their manufacturing and production processes. The human resource management applications maintain data on over 500 employees and more than 60 percent of their employees (> 300 employees) required and were given access to one or more computer systems and applications that were part of the IT infrastructure at ISI Inc.

The IT infrastructure at ISI Inc. included a variety of computer systems and applications. Some of these systems were integrated together in a networked environment while others were standalone. The corporate applications (CAS) resided on a Microsoft Windows NT-based client server network that included a set of desktop computers running Windows NT. The customer data and customer service applications are maintained on a system of Oracle databases that we shall refer to as Customer Service System (CSS). The production data is also captured on a different system of Oracle databases and we will refer to this as the Production Data System (PDS) for convenience. The manufacturing applications and data are managed by a manufacturing application system (MAS) installed on an IBM AS/400. The human resource (HR) data is maintained in a spreadsheet application (MS-Excel) and is part of the data that resides in the NT network for corporate applications. Each system is part of a specific functional unit within the organization. The manager of a functional unit is responsible for authorizing and managing the users in the system that is part of that functional unit.

The user-management was performed at two levels: the system level and the application level. The hardware/platform specifically, the operating system for each hardware/platform controlled the access for each user at the system level. This defined the authentication of the individual users, the files and directories within each system that the user had access to, and the read, write, and execute privileges each user held for the different applications that resided on this system. This is accomplished by creating groups/roles, assigning a set of permissions and privileges for each, and enrolling each user as being part of the appropriate group/role. The application level user-management managed user access to the specific application modules and associated queries (and data). It defined the manner in which each individual user accessed each module, defined the types of permissions (read or write/update) each user had on the data associated with each module, and defined the specific view of data for that user or the set of users. This was accomplished largely through the creation of user groups/roles (similar to the system level) with the exception of the MAS.

The MAS application system included 19 application modules. Each module performed several functions such as querying accounts payable, maintaining accounts payable master, identifying payment cycles, and reporting receipts that are not invoiced. These functions may be part of a single application module. As each module performed several functions ISI Inc. managed users by providing them access to specific functions within each module. To accomplish this, each module had several groups, each group with the privileges to execute a specific set of functions. A user was assigned to a group depending on the function(s) he/she needed access to. Each user was part of many different groups in different modules. The MAS hence had 277 unique user identifiers but over 9500 records to capture the user-access data to the different groups and application modules. The number of groups in each application module ranged between 6 and 50 (with an average of 27). There were over 500 such groups distributed over 19 modules. The number of users in each group varied between 1 and 210. There were 19 groups with more than 100 users in each, 24 groups with 50-99 users each, 191 groups with 10-49 users in each, and over 270 groups with less than 10 users in each group.

Users in the CAS were managed using 108 groups. There were 482 unique user identifiers created in the CAS belonging to the 108 groups defined. The user records in CAS numbered more than 2500 as each user was assigned to several groups. Interestingly, 16 of the 108 groups created did not have any users, 3 had over

380 users in each, 5 with 50-99 users in each, 34 with 10-49 users in each, and 50 groups with 1-10 users in each. The PDS was accessed by 19 unique users and managed with 5 groups (one with 10 users and the other 4 with 2 or 3 users in each) and each user belonged to one and only one group in this system. The CSS users were divided into three categories based on how they used the system: the developers, the database users, and the CSS application users. The 5 developers were assigned to one group and their privileges managed using that group. The database users (22 users) were managed using 3 groups. The administrator of the Oracle database was the only member of the administration group. Two experienced Oracle users were allowed to create ad-hoc queries from any of the tables in the database and were part of the second group. The rest of the database users belonged to the final group that permitted them to run pre-defined periodic reports. Members of the first and second group were also part of the third group. The CSS application users were managed using 10 groups. Some of these groups are customer support, sales, administrative groups for each, system administrator, training, and a group for supervisors. Of the 86 application users, 63 belonged to the customer support group.

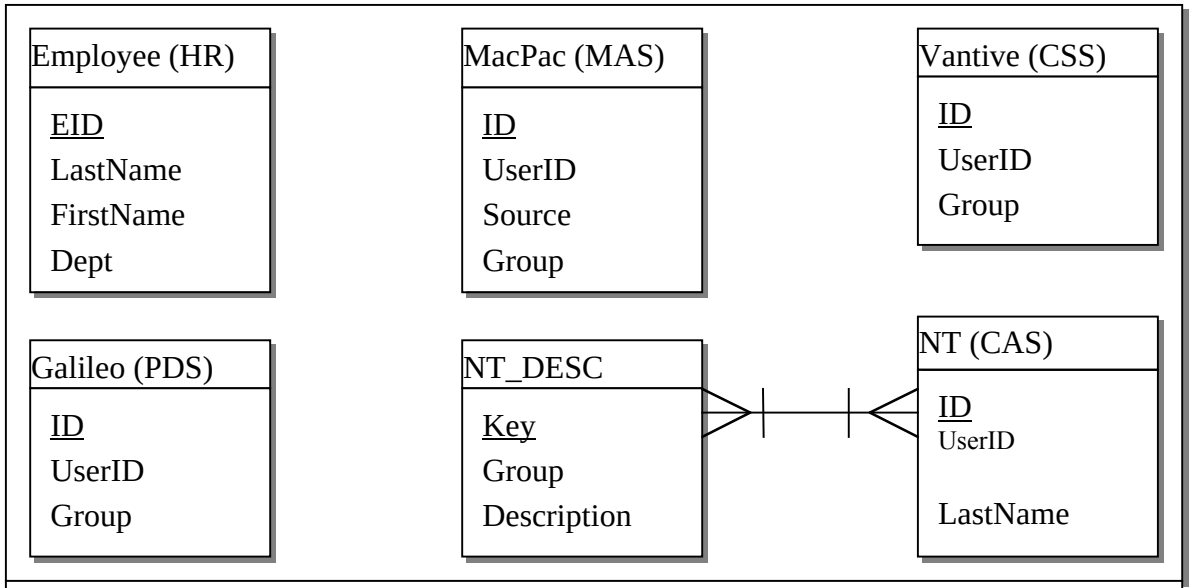

Figure 1 Original Schema

Employee information was maintained and managed by the HR department. Each employee of ISI Inc. was permitted to have multiple user identifiers and the first problem that ISI faced was matching employees with their corresponding user identifiers. Many of the user accounts were no longer used and these accounts continued to remain in the different systems at ISI Inc. Also, HR did not have employee information for many user accounts since some accounts were given to contracting employees who were not maintained in the HR database. These problems surfaced when the managers of functional units in ISI Inc. were asked to verify each of the user accounts and the access these accounts had on their systems. The purpose was to ensure that the right employees had the necessary access to the system functions and data and to remove the defunct or unauthorized accounts. No coherent report on users of systems could be generated using the existing data on user accounts. The management discovered that integrating the set of users across all of the systems in ISI Inc. was a complex task. ISI Inc. also understood that this situation compromised information security at ISI Inc. The specific objectives in integrating the user accounts included the following:

1. Reconciling employee names with user accounts to identify the employees (and contractors) who owned user accounts on the systems.

2. Identifying the groups on systems and modules in systems and understanding the privileges

given to each of these groups. Almost all systems had multiple distinct groups with the same set of privileges but were managed separately. Users were assigned to one or the other and some users belong to both groups.

3. Identifying users of each system and the groups within the system that the users belonged to. This aids in better managing each user with a single user account and reduces the "garbage" in the user management data.

3. Designing a Database for Integrated User Management

The organization had four systems, each maintaining their own user accounts: a Windows NT 4.0 domain, and three mainframe systems: Macpac, Vantive, and Galileo. In addition, there was a database, unrelated to the user accounts databases on any of the systems, that stored employee information, such as employee IDs, names, and so forth. The schema we started with is depicted in Figure 1. The problem was that we had to answer the question: who has user accounts on which system, and what groups are they members of? None of the systems could provide us with that information, and of the four, only Windows NT stored anything that could be used to relate UserIDs to employee names.

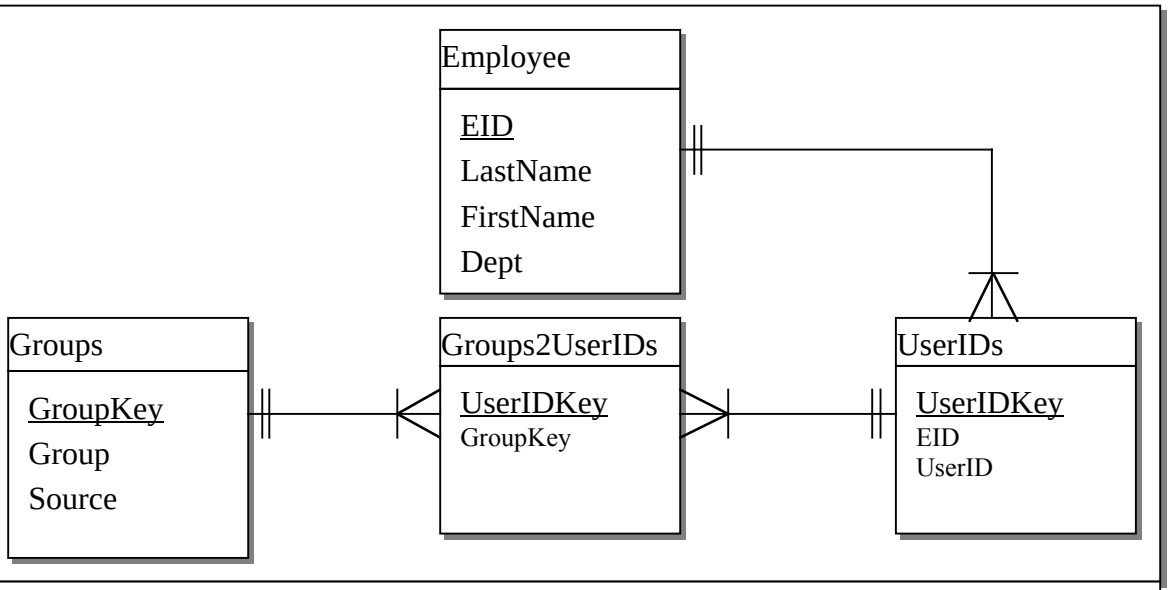

Figure 2 Optimal Schema

4. Removing defunct accounts, and more importantly, removing accounts for employees who were no longer with ISI Inc.

5. Communicating to each user the specific manager(s) he/she should report to when accessing restricted data in each system.

6. Generating reports on users to assist managers in identifying and authorizing the users of the system that the manager was responsible for.

We started by defining the optimal schema; essentially stating what we wanted to know. Then the task was to fit the data we had into the schema that we wanted to produce the result we wanted: a list of employee names that listed their usernames and access to all systems.

The optimal schema is shown in Figure 2. The problem, however, was how to migrate the data from the original schema to the optimal one. The additional data in the Windows NT account database presented us with an avenue. We therefore used NT to create the link between the employee table and the UserIDs table.

The first step was to remove all duplicates from the NT table, and then join it onto the Employee table to map the NT usernames to employee IDs. This was done using a right join of Employee onto NT. The right join ensured that all NT usernames, even those that did not match an employee ended up in the table. The problem with this approach is obvious: Employee first and last names are very often not unique. To deal with that problem we put the data into a temporary table, called NT_Users, which we could analyze for duplicates later.

To match up usernames to employee IDs, a method for relating usernames to employee IDs is needed. With the Windows NT accounts, we can assume that if the first and last name of the NT user matches a first and last name in the Employee table, we can relate the NT username to the matching employee ID. However, the remainder of the systems do not have any name information, other than the UserID itself. We therefore need to make the following assumption:

Assumption 1: *If a username in any of the other systems matches a username in the Windows NT database, they refer to the same user.*

Assumption 1 sounds very logical. But, it is quite a leap of faith. However, in the situation where the systems have grown uncontrollably without a central authority, such as the one presented here, it is the best we can do. Since the employee database contained a manager for each employee, a business decision was made to create a list of user accounts for each manager and make the manager sign off on the list and report any anomalies, such as employees that no longer work for the organization.

Under assumption 1 we can now go ahead and match user accounts in the various systems. We do this by using a left outer join of the appropriate table with the temporary table created to hold the NT Users. The join condition is the UserID. This ensures that all those instances of users that do not match a UserID in the NT Users table get included. For those that do match the IDs in the NT Users table we also add the EID. Doing that for all the other systems results in a set of new tables that have the appropriate UserIDs for each user, as well as an EID where available.

Certain UserIDs did not match a known entry in the Employee table, and consequently no EID could be automatically assigned to those entries. A separate table called MISMATCH was created to highlight these entries. The MISMATCH table contained only three attributes: the UserID, the EID (which started out blank), and the source of the entry (the system it came from, such as NT, Macpac and so on).

The entries that end up in the MISMATCH table essentially fall into three categories:

1. Contractors who are not regular employees of the corporation and therefore do not have an EID, or even an entry in the Employee table

2. Names where the entry in the NT Users table do not match that in the Employee table, such as Jim versus James

3. Employees who changed their names for some reason, such as a marriage

Of these, numbers 2 and 3 are relatively easy to deal with by inspecting the entries and comparing them to entries with the same first or last name in the Employee table. This requires two queries, one that joins the MISMATCH table onto the employee table based on first name only, and one that joins the MISMATCH table onto the employee table based on last name only. Historical versions of employee tables help greatly in identifying entries falling into category 3. In the end, those entries that could not be resolved in this manner were passed to a manager to authorize. This was done for those entries where uncertainty remained as to the true owner of the UserID

Contractors presented a special challenge. These entries do not exist in the Employee table, for several reasons. Therefore, it becomes a challenge to locate the person and that person's manager to authorize the account. In section 5 some strategies for using Windows 2000's Active Directory for better tracking these users are presented. However, in most cases, it holds that it is imperative to track contractors in some form of database. This could be either the same or a similar format database as what is used to track employees. If contractors do not have an EID, one should be assigned to them that then links to the employee table such that a manager can be tracked for that contractor.

Once entries in the mismatch table have been assigned IDs, those entries can be re-introduced into the tables holding the users from the particular system.

After verifying that all UserIDs in the various user tables have EIDs, we can export them to the UserIDs table, and turn to dealing with the groups.

To build the Groups table is not terribly difficult, given that the original User tables have a group for each person. The problem is that the tables are not in any kind of normal form. Hence, if an employee is a member of several groups, that employee had several entries in the user table. A simple SELECT DISTINCT from each users table solved the problem and populated the Groups table with all the groups. As part of each SELECT query from the users table the source system for each group was populated as well, using a predetermined string.

At this point, three of the four tables in the final model are finished. The only one remaining is the intersection entity Groups2UserIDs, which holds the primary keys from the UserIDs and Groups tables and maps UserIDs to the groups they belong to. Creating that table, again, is not difficult. A single query can be constructed for each source (system) that joins the user IDs in the UserIDs table to the original IDs in the source table, extracts the group information, and inserts this into the Groups2UserIDs table.

4. Tracking Better Information

One of the major problems faced when attempting to validate users is that of incomplete information. Most systems do not allow you to track any information about user accounts, other than the account itself and any groups that the account is a member of. However, as we saw in the discussion above, this leads to problems when attempting to identify the users. Without a reliable way to tie a user account to a person, administrators cannot be sure whom the accounts belong to and who is responsible for the security and proper usage of that account. A very basic security requirement is that the system is capable of holding users accountable for their actions on the system. However, holding user accounts accountable is not useful. For accountability to be meaningful, we must be able to map user accounts to a physical person, which can be held accountable for the actions taken with that account. In addition, if a person no longer needs the account, we would need the person's manager to decide whether to disable the account. This information, as well, must be tracked by a system tracking user accounts. As we saw above, only Windows NT had some ability to track that information, in that it can track a person's name. However, even Windows NT fails in its ability to reliably connect that to an entry in an employee database. None of the other systems had even that basic functionality.

In February of 2000, Microsoft released the long awaited upgrade to Windows NT 4.0. Originally named Windows NT 5.0, the market name for the product at release is Windows 2000. Windows 2000 is quite possibly one of the most complex products ever designed by man, in any category. One of the features that Microsoft added to Windows 2000 is the ability to track more information about users. Windows 2000 is built around a directory services module known as Active Directory. The Active Directory is essentially a database of objects that exist in an organizational computing environment. It can track not only user accounts but also computer accounts, groups of users and computers, organizational groups (known as Organizational Units) and many other objects. Microsoft's objective was to supplant the organizational employee database with Active Directory. Whether they have succeeded is a matter of significant debate. What is certain, however, is that the Active Directory can track much more information about each user than the SAM database in Windows NT 4.0 could.

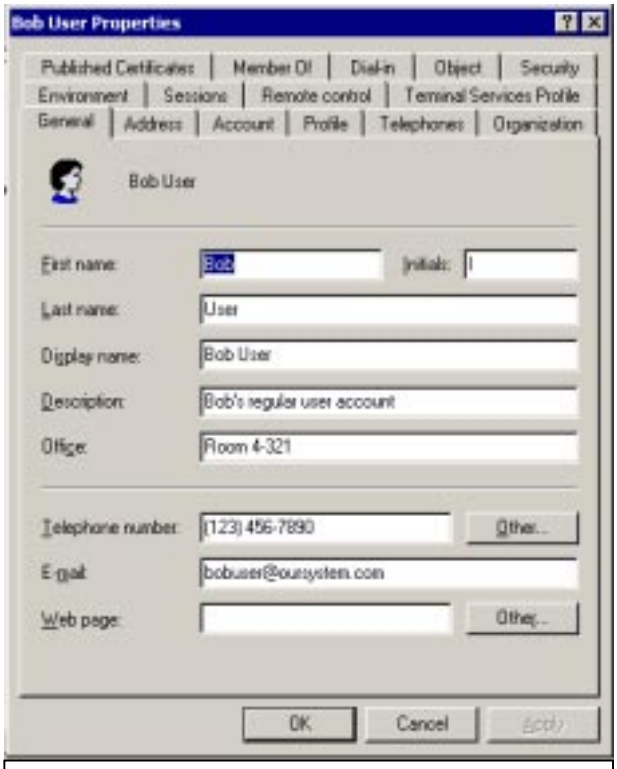

Figure 3 User Properties Dialog

Figure 3 shows the user properties dialog in the Active Directory. It is possible to track not only the user's name and a description, but also telephone number, office number and e-mail address. Furthermore, by clicking the Organization tab, one is presented with the screen shown in figure 4.

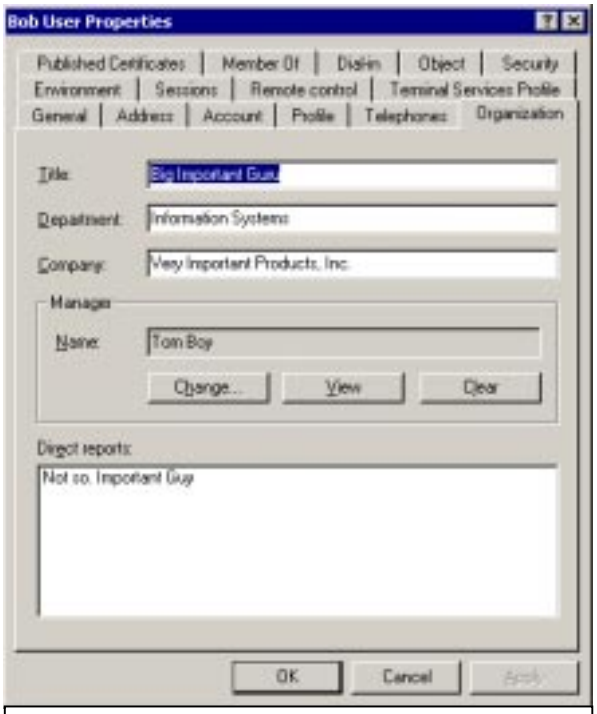

Figure 4 Organizational Information Tracking

Active Directory permits tracking of organizational attributes, such as the person's title, the department they are with, and the manager. These are all items that were stated as missing from the systems involved in the project described above, and are a significant advance. It is also possible to interface with the Active Directory through a set of APIs that can be used directly by a programming language, or through a data provider, such as the ActiveX Data Objects (ADO). This affords the ability to create an application that simultaneously queries a SQL database that serves as a repository for user accounts from a different system and the Active Directory, and presents this information in a uniform manner.

This means that the Active Directory can be conceived of as a node in a federated database system, and treated as a source by an application that provides services such as a repository of users. The reporting capabilities are extended significantly over what is possible with the information present in the systems described above. However, this information needs to be tracked in the Active Directory to be useful. To enable an organization to track that information, a procedure such as that outlined above is necessary to identify the existing user accounts. Few organizations have the luxury of starting over with their user accounts databases today.

There are several very important pieces of information missing in the Active Directory and assumptions that were made in its design. The first one is that an employee only reports to one manager. This is not necessarily true. Many organizations employ a matrix structure where people have multiple managers. In addition, certain employees are "lent" to other managers on a project-by-project basis. These employees may have a permanent manager and a temporary manager, each one of which might need to be notified of events regarding that employee. Furthermore, category three of the users found above to have no Employee ID were employees that had changed their names, e.g. as a result of a marriage. Therefore, it may be desirable to track employees' former names to be able to match user accounts on different systems to those on Windows 2000.

All of these values can be added to the tracking capabilities in Active Directory by means of extending the schema of the Active Directory. The Active Directory schema is designed such that it can be extended with new types of objects and attributes. This can be done through the capabilities of Active Directory itself. However, schema changes are irreversible operations in Active Directory. While new classes and attributes can be disabled, the changes cannot be undone. Therefore, it may be preferable to create a middleware application that joins the Active Directory to a database that contains the additional attributes. Further research is needed into this technique to determine what is the best course of action. Depending on the operations needed, schema extensions may be the only viable solutions.

5. Conclusion and Further Work

In this paper we have described a typical problem that organizations face - that of managing user accounts belonging to the many different IT systems in the organization in an integrated fashion. Organizations are increasingly forced to deal with this situation due to the emergence of Intranets / Extranets, the need to share data/applications both within and across organizations, and the need to migrate to new systems combined with the desire to track more information about users on

those new systems. Ad hoc creation and management of user accounts to support Intra/Extranets results in the wasteful duplication of user information. More importantly, managers lack the ability to identify and associate employees with user accounts and match user accounts with system/application privileges granted to each. This compromises the security of their corporate network of systems and must be resolved. In this paper we have examined one such situation and proposed a practical implementation to address the resulting problems. The proposed solution offers several benefits besides being simple and practical. It is not specific to managing users on a particular system and can be used in an integrated environment with several types of systems as illustrated here. Further, it is scalable to include applications and networks that users may be authorized to access and hence requires user management. Finally, it can be implemented in any relational database system or incorporated into the Active Directory of MS-Windows 2000 to exploit the user accounts and related user information that is already tracked by that database.

In extending this work, there are two specific research directions that need to be investigated. The first is to examine the capabilities of the Active Directory with the view to extend our solution to include the capabilities in this user management tool. Active Directory performs extensive user management services but is restricted to the Windows 2000 network. Organizations typically maintain several networks of systems and managing all of these using Active Directory is impossible. The current solution described here has a very limited set of user management capabilities when compared with those offered by the Active Directory. The objective of this extended solution is to provide organizations with the ability to comprehensively (all user management services) manage user accounts across all networks in an integrated fashion. An application that integrates the Active Directory with the user account databases of several other systems is therefore needed. This application could store additional data in a database of the administrator’s choice, and have the ability to act as the interface to all of the managed systems. Such an application is under investigation by the authors of this paper.

Each individual operating system (network or otherwise) and application supports the management of users on that system/application. One approach to integrating several such systems/applications/networks for integrated user management is to model user information in each system as a simple relational schema. The resulting set of schemas can be integrated to create a federated/global schema by applying well-known techniques for schema integration. This type of schema is a requirement for the management system described above. Several questions need to be answered before realizing this solution. Would the federated schema accurately track user account information? How scalable or extendible is this schema and how would changes to individual schemas be reflected and managed? What is the overhead cost/time involved in creating and managing such a federated schema? Attempting to answer such questions and defining a global solution for integrated user management is the second research direction that is currently under investigation.

References

Department of Defense, "Department of Defense Trusted Computer System Evaluation Criteria," Report DOD 5200.28-STD, 1985.