

Solution: Authorization Data Model

The subject of the invention is a way of modeling authorization information that dramatically reduces the number of roles, which need to be maintained in a role-based system. The core idea is to separate the role, reflecting the position of an individual in an organization, from the responsibilities, reflecting the areas of activity, which need to be accessed by this individual.

The simplest reflection upon the system is that the information about the actions an individual is allowed to take are separated from the information qualifying the data upon which these actions are taken.

The general structure of existing authorization management systems is composed of individuals, roles, profiles or groups that qualify these individuals and authorizations/privileges that belong to the individual. FIG. 1 shows diagrammatically the basic design of the typical authorization management system. In this system the individual who is being checked for authorization to access a protected resource is presented as a query to a database maintained by the application, which, using the individual's role within a group and its associated privileges, decides on the basis of the fixed role and privilege information whether a particular action is authorized or not. The result of this determination is the value, for example yes or no, reflecting whether access is or is not granted, or conditionally granted in some cases.

This leads to the implementation of the data model in a typical relational database as shown in FIG. 2. The fields consist of user, role, object, action and value. The user field contains the user ID of the individual. The role field contains the specific position information, the object field identifies the data for which access is being evaluated and the value field is the authorization status, i.e., access granted or not granted. The data model of FIG. 2 is not only static or fixed, but it is replicated in multiple applications and user authorization tools.

Performing the same actions, or having the same access rights for different objects, results in different authorizations. In order to differentiate, one will need to create different roles to reflect those changes. Imagine the simple example of one company giving the sales people the right to access the product sales information in their own area. For each and every area, there will be another sales role created for that company.

With the model proposed by this invention, the fields are not static anymore, but include generic variables. The input for filling the fields is requested at run-time from an external system containing information about the user, and/or the fields of the record that are already filled at the time of the enquiry.

This can be for example realized through a backtracking link as shown in FIG. 3. The static authorizations composed of the elements object, action and value are replaced by authorizations composed mainly of wildcards for the elements action and value. The object will be given at runtime by the application determining the access rights.

The system relies on decomposition or factoring of positions into two basic components: roles and responsibilities.

Objects are "protected resources" and could be data or a transaction or instantiation of a class, etc. In the context of authorization, an "object" is a protected resource that the user wishes to access in some way. The "action" is the requested manner and extent of access that the user seeks with respect to the object. So if the object is data in a database, the action could be read, write, edit, move, or delete. If the object was a particular screen, the action could

be display. If the object was a transaction, the action could be approve, reject, comment, review, complete, and so on.

FIG. 3 shows a system that partakes of both the prior and the new system. As in the prior system, the user ID would be paired with permissions through a static mask retrieval process, as shown in FIGS. 3 and 4. So knowing the ID and the object and action to be applied to the object, the same database could be queried for a yes/no value for a given action. This information is static in the authorization database. If the action/value fields have not been filled in already by the static system and need updating, the new system, according to the invention, triggers a look up of the meaning and implications of the individual's roles and responsibilities in an external system. This dynamic authorization decision triggers a call to a different database where the rights and privileges associated with these roles and responsibilities are stored. So the privileges associated with roles and responsibilities are disassociated from the individual per se. Rather they are associated with generic roles and responsibilities that may be changed and supplemented and updated from time to time without reference to or even knowledge of a particular individual. The difference between these two systems is that in the static, fixed data model, the fields are fixed, and in the dynamic system, they are completed at run time.

FIGS. 5 and 6 merely show conceptually the portion of the prior system that is addressed and replaced by the new approach. The grayed blocks containing static information on role/group, authorization and value are basically replaced by the wild card model.

One example of this dynamic authorization decision process is shown in FIGS. 7 and 8. Here the dynamic decision process of FIG. 3 is implemented by way of a query or call to an external database which holds the responsibilities and associated permissions for various positions and cost centers in a company, Based on the content of the external database, at run time, which may be different from time to time, the action/value fields are filled dynamically resulting in the authorization.

Another visualization of this concept is provided in FIG. 9., showing interaction between one or more applications with a central authorization repository. Here an application with access to a protected resource, e.g., data or a transaction, receives a user request for access to the protected resource. The application contains an authorization module or tool that generates, in response to the access request, an inquiry to a central authorization repository maintained for example by the system that hosts the user information database within the enterprise, e.g., human resources, project planning, or CRM (customer relationship manager) for customer links. This repository, which may be a data storage system implemented in the form of a relational database, contains a plurality of role definitions, in the form indicated as role No. 1, role No. 2, etc. These roles are not uniquely associated with a particular individual but are available for assignment to any individual. So for example, in FIG. 9, users 1 and 2 are assigned the same role 1, e.g., sales manager, but have different respective responsibilities 1 and 2, e.g., Argentina and Brazil. Similarly users 1 and 3 have different roles, e.g., sales manager and technical writer, but the same areas of responsibility, e.g., Argentina.

The meaning or definition and associated privileges are defined by a role definition management system, through which the description and functions of the roles can be manipulated and redefined from time to time, independent of the identity of the individual having those roles and independent of changes in personnel. In addition, the repository