MIT Authorization Management

Jim Repa and Paul Hill

May 2, 2008

Infrastructure Software Development and Architecture

Information Services & Technology

Massachusetts Institute of Technology

The Roles Database at MIT

- Working system, has been in production at MIT since 1998
- Controls access at coarse and fine granularity within various systems and applications at MIT, including
  - SAP financial (spending, approving requisitions, approving travel documents, approving invoices, reporting, etc.)
  - HR (various HR transactions, reporting)
  - Student Systems (graduate admissions, undergraduate admissions, registration, and financial aid)
  - Environment Health and Safety (training reporting, laboratory space recording,…)
  - Central services such as the Warehouse, MIT ID system, Master Department Hierarchy, Roles Database, …

Guiding Principles

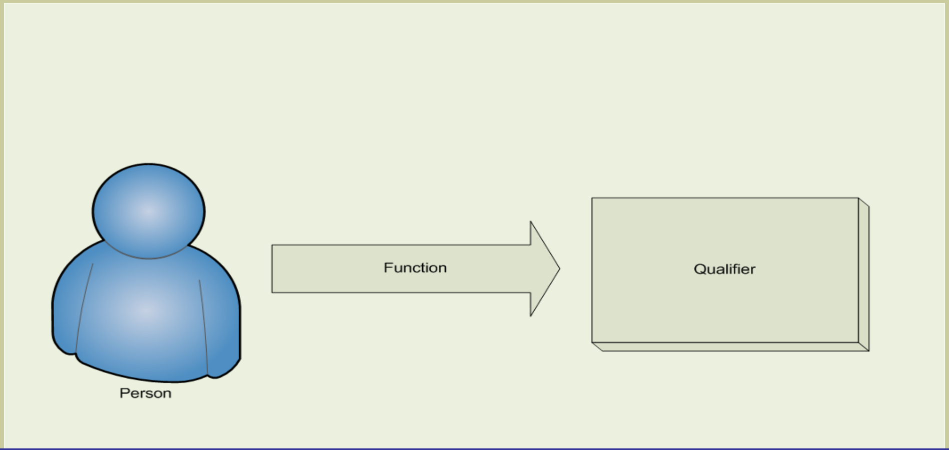
- Central authorizations repository - feeds data to other systems, which in turn enforce the authorizations
- Authorizations are defined in understandable business terminology, not arcane technobabble of each system. Define "functions" (transactions or roles for which authorizations should be assigned) at the appropriate level
- Maintenance of authorizations are distributed to departments, labs, and centers - keep the maintenance activities close to the people who understand the business needs.
- A single authorization can feed more than one system, e.g., financial reporting authorizations control access to reporting both in SAP financial system and in data Warehouse

New rules-based enhancement - principles

- Use rules-based system (under development) for assigning "implied" authorizations based on known attributes about people. This component complements, but does not replace the existing system for assigning authorizations to specific individuals

- Decide where rules are appropriate, but do not overuse them. Rules work for access to some resources (e.g., access to Library materials and downloadable licensed software) but individually-assigned authorizations are appropriate for other resources (financial and HR transactions)

3 Part Authorizations



Examples:
Joe + Can Access + Oxford English Dictionary Online
Jane + Can Download + MS Office 2007
John + Can Modify Voice Mail Forwarding + 6172589850
Jerry + Can Create Functions + in category HR
Juan + Can spend and commit + on cost object Q678543

Simple building blocks, complex behavior

- Categories contain functions
- Each function is associated with a particular type of qualifier
- More than one qualifier type can exist within a category
- An authorization has a start date
- An authorization may have an end date
- A person may have the ability to grant an authorization to others
- A Function can have child Functions. (inheritance)
- Qualifiers are organized into hierarchies. (inheritance)

How authorizations are distributed and enforced

Two different models employed by various applications:

1. Extract appropriate authorization information periodically, either through Oracle database queries or other methods, and cache the data within the application's own database

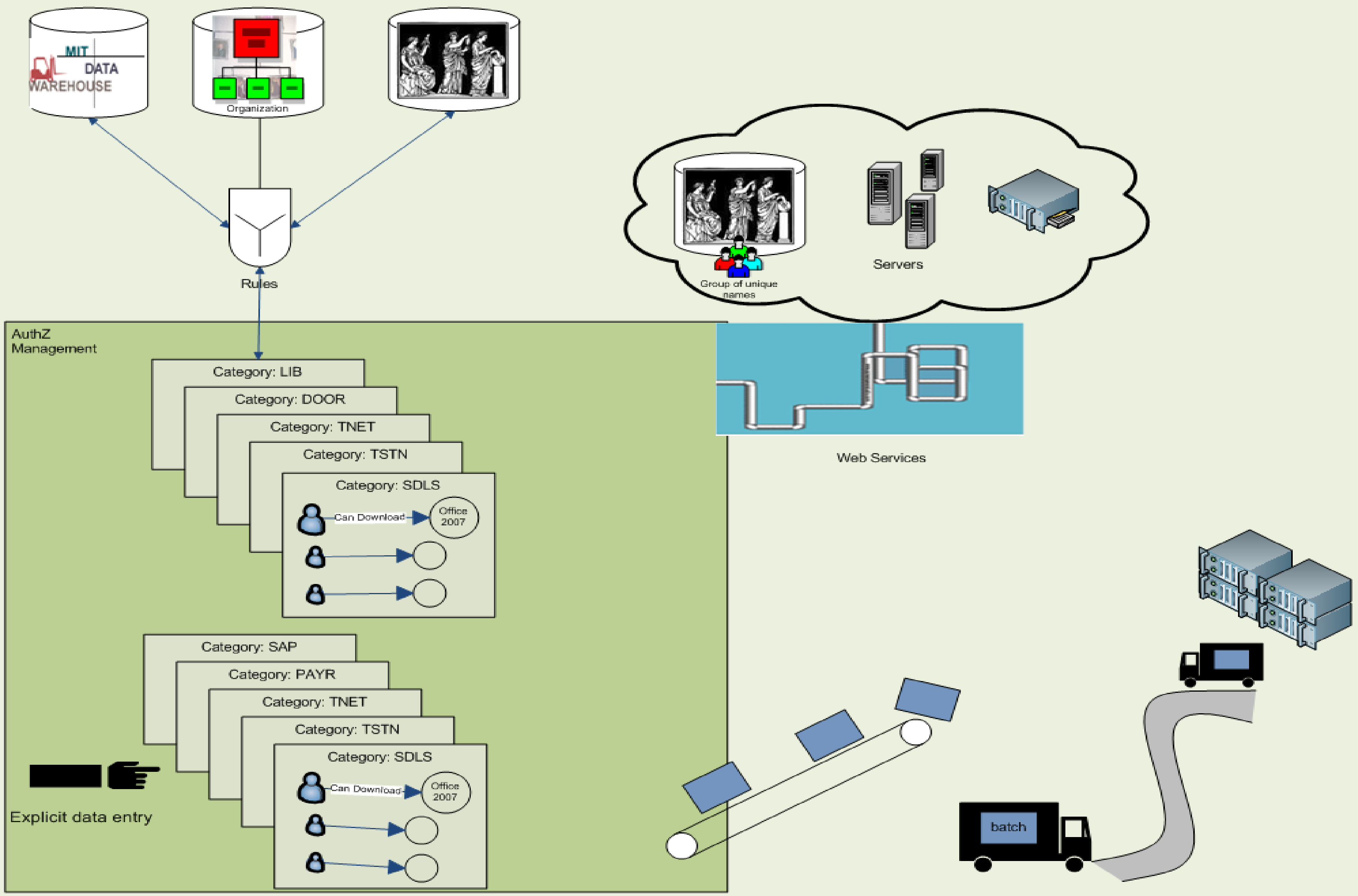
2. Look up Authorizations in real time using a web service, direct connection to the Oracle database, or other interface (Note that shadow tables and indexes are structured so that authorizations can be looked up quickly, without having to do a recursive tree traversal every time we look up an authorization.)

Why 3-part authorizations and not simple groups or attribute pairs?

- Authority to do various transactions naturally falls into 3-part structures.
- Reporting, spending, hiring, downloading, updating data, and other activities naturally has a "verb" part and an "object" or some limited area or set of objects where the person is allowed to do the transaction.
- Membership in a group hides or ignores the Function+Qualifier nature of most activities that are controlled by authorizations or permissions
- We can think of the set of people who are allowed to perform function F with qualifier Q as a virtual group. Within the Roles DB at MIT, we don't actually define an object to represent that group, but one could think of it that way.
- When you define Functions and separate Qualifiers, it is easier for an Auditor or administrator to review the permissions of the people who own Authorizations than it is if you're just given a list of groups with group memberships.
- Groups are not semantic. The meaning at best may be captured in the name if a naming convention is used. Remember, naming is hard.

Why not more than 3 parts?

- Our experience is that almost all permissions can be broken down to Person + Function + (single) Qualifier.
- Too many qualifiers usually means that you're not thinking of Functions at the right level of abstraction.
- A few hypothetical counter-examples have shown up in our actual use cases. (We are planning a small extension to accommodate the needs of this small number of cases that needs another dimension.)


MIT DATA WAREHOUSE
Organization
Rules
AuthZ Management
Category: LIB
Category: DOOR
Category: TNET
Category: TSTN
Category: SDLS
Can Download
Office 2007
Category: SAP
Category: PAYR
Category: TNET
Category: TSTN
Category: SDLS
Can Download
Office 2007
Explicit data entry
Group of unique names
Servers
Web Services
batch

Data modeling for implied authorizations

- Design a rules facility that is simple enough to build and maintain without years of development, yet versatile enough to cover reasonable set of use cases
- For "conditions" (attributes) that determine who is included in a rule, use data organized into 3-part "relations", with a subject, verb/connector, and object, e.g.,
  - ("Username JOEUSER", "Is a current student", "Department of Biology")
  - ("Username FREDUSER", "Is a current employee", "Dept. of IS&T")
  - ("Username SUE", "Has been certified", "Training xyz: Use of radioactive materials")
- Each part of the triplet represents a (i) person, (ii) previously defined function/relation/connector, (iii) previously defined object of a type that matches the function/relation/connector component.

Rules have 4 main fields.

Current model-in-progress identifies 4 types of rules. Each rule (regardless of type) has 4 main parts:

- Conditional function/relation/connector
- Conditional object or branch of a tree of objects
- Implied function (what transaction a person will be permitted to do)
- Implied qualifier (area or object for which the person is permitted to do the implied function)

Usage of rules

- Rules will be evaluated periodically (nightly or more frequently) to create implied Authorizations
- Consuming applications will use an interface or web service pointing to the middleware component to ask questions about a person and his/her implied Authorizations.
- The evaluation of rules will be handled by the middleware and backend database, and the consuming application *does not need to know about the prerequisite conditions or rules.*
- The consuming application does not need to know whether the actual implied authorizations are physically stored in a database or built on-the-fly.

Usage of rules (2)

- A UI will be made available for specially-authorized administrators (in each application area) to maintain rule definitions.
- An interface will also be built (web service or other) so that a consuming application can integrate rule-maintenance with its other functionality.

APIs (SOAP/WSDL), done

- isUserAuthorized – (username, category, function, and qualifier_code) return a TRUE/FALSE answer
- getUserAuthorizations – (username, category, [function_id]) return a list of authorizations for the person.
- createAuthorization (username, category, function, qualifier_code, start_date, expiration_date, […])
- updateAuthorization (authorization ID, fields to be updated)
- deleteAuthorization (authorization ID)
- listFunctionCategories () Lists existing Function Categories

Planned APIs (SOAP/WSDL)

- createFunction Given the components of a Function, create a new Function.
- updateFunction Update one or more fields for an existing Function.
- deleteFunction Delete an existing Function.
- addFunctionParent Add a parent/child connection between two functions.
- deleteFunctionParent Delete a parent/child connection between two functions.
- createFunctionCategory Create a new category for functions
- updateFunctionCategory Update the description or other attribures of an existing category

Planned APIs (2)

- deleteFunctionCategory Delete a category for functions
- createQualifierType Create a new qualifier type, along with a root node.
- deleteQualifierType Delete a qualifier type.
- createQualifier Create a new Qualifier, including its connection to a parent Qualifier
- updateQualifier Update one or more fields for an existing Qualifier
- deleteQualifier Delete an existing Qualifier
- addQualifierParent Add a qualifier parent/child relation
- updateQualifierParent Attach a qualifier to a different parent

Planned APIs (3)

- deleteQualifierParent Delete a qualifier parent/child relation (must not be the only parent of the child qualifier)
- addQualifiertype Add a new Qualifier Type
- updateQualifierType Updates attributes of a Qualifier Type
- deleteQualifierType Delete a Qualifier Type
- addImpliedAuthRule Add a rule for implied authorizations
- deleteImpliedAuthRule Delete a rule for implied authorizations

Info about APIs

- **Web Service interfaces to the Roles Database - http://web.mit.edu/repa/www/roles_ws1.html**
- https://authz.mapws.mit.edu/uaws/services/ua?wsdl
- https://map-test-ws1.mit.edu/rolesws/services/roles?wsdl

References

- MIT's Roles Database: Parts of an Authorization - http://web.mit.edu/repa/www/roles_auth_parts.html
- Storing and Presenting "Relations" Data at MIT - http://web.mit.edu/repa/www/relations_db_notes.html
- MIT Roles - http://roles.mit.edu/
- Roles web application - https://rolesapp.mit.edu/rolesclient/rolesui.html# (requires authentication)

Questions?

The end

Extra slides

Ignore the man behind the curtain

An "Authorization" consists of three parts:

1. a person
2. a function
3. a qualifier (an organizational, financial, or other unit which defines the area, object, or set of objects where the person is authorized to perform the function)

How authorizations are structured

- Person + Function + Qualifier
- When creating or modifying an Authorization
  - pick person from a pre-existing list (loaded each night from warehouse)
  - pick Function from a pre-existing list
  - pick Qualifier from a pre-existing list
- An Authorization also has some other fields, including start date (usually same as the day it was assigned), end date (usually null, i.e., Authorization remains in effect until it is explicitly deleted), grant flag (if turned on, the owner of the Authorization can assign it to others)

Functions

- Functions are grouped into Categories (where each category represents an application area). Functions are maintained within the Roles Database by a small set of administrators. (Many people can maintain Authorizations; very few can maintain Functions).
- A Function can have child Functions. (inheritance)

Qualifiers

- Qualifiers are of various types (e.g., HR Org Units, financial Profit Centers and Cost Objects, academic majors, telephone numbers at MIT, etc.).
- Each Function is associated with a specific Qualifier Type. When you create an Authorization and pick a Function, you see a pick-list of only the appropriate Qualifiers, which are (i) of the appropriate type and (ii) are available to you based on your authority to create Authorizations.
- Qualifiers are organized into hierarchies (not strict hierarchies, since it is possible but rare for a Qualifier to have more than one parent). (inheritance)
- Most Qualifiers are fed from other systems (e.g., HR Org Units, financial hierarchies of objects, laboratory "room sets" and individual rooms). A few Qualifier types are maintained via an administrator's UI within the Roles DB itself.