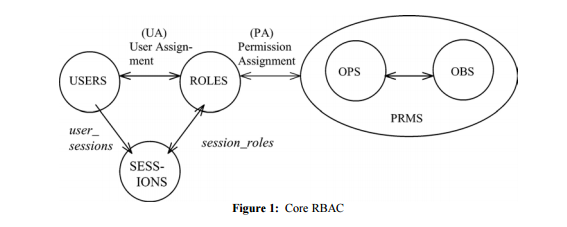
Security Use Cases

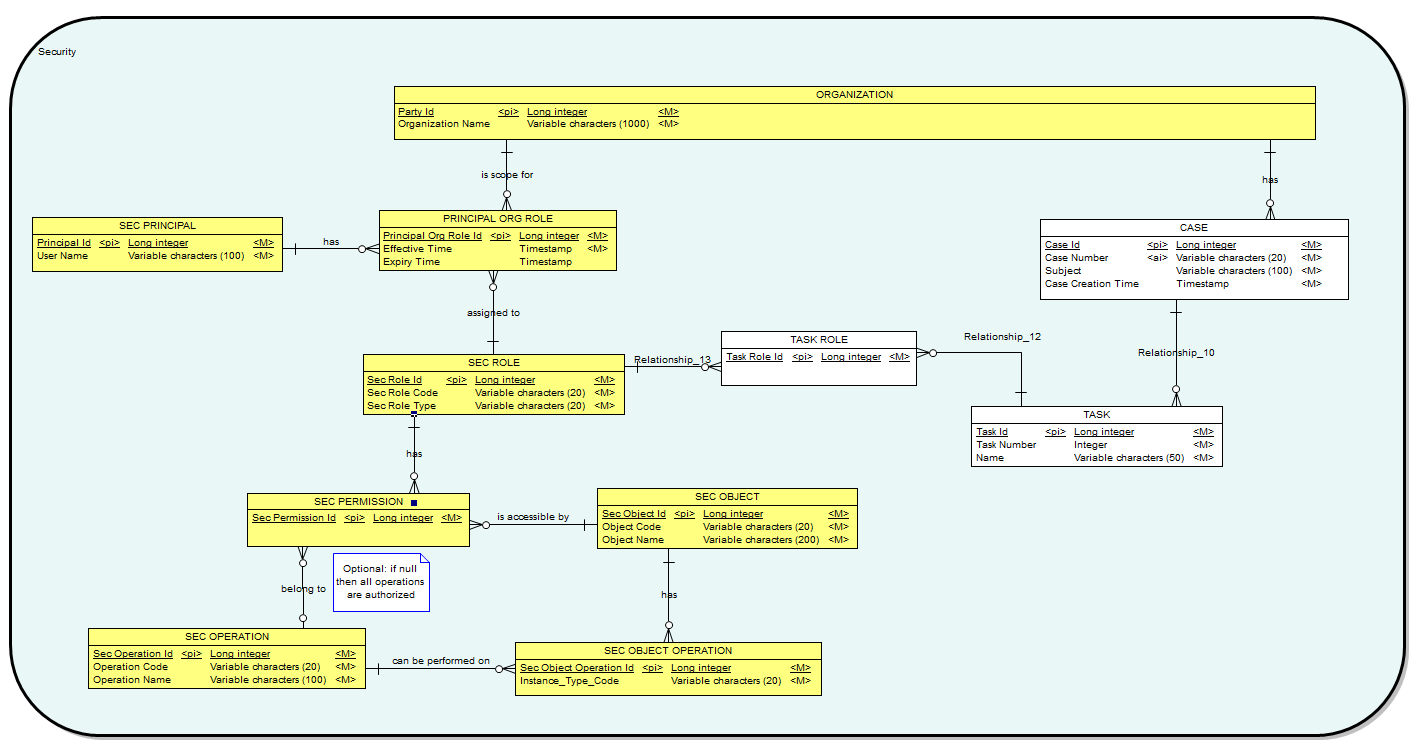
# RBAC

Our data model implements a data store for core RBAC (except for SESSIONS but including the extra wrinkle of adding organization scope). Here’s a picture of core RBAC:



# Data Model

DDL for this model is attached.



# Entity Definitions

|  |  |
| --- | --- |
| **Entity** | **Description** |
| PRINCIPAL | User or Group of Users. In POC just User |
| ROLE | A collection of permissions (see below) |
| OBJECT | An object for which there can be one or more operations |
| OPERATION | An operation that can be performed on an object |
| PERMISSION | Represents permission to perform an Action on an Object |
| OBJECT OPERATION | The operations that are valid for each object. |
| ORGANIZATION | Provides the scope for which a Principal is associated to a Role |
| PRINCIPAL ORG ROLE | A Principal is associated to a role for an organization. |
| TASK | A step in a workflow. Initially assigned to one organization plus one or more roles. |
| CASE | The thing we want to perform actions on and that the system needs to secure. |

# Test Data

Test data is found in file permissions.yaml. This file contains data that can be used to populate all the tables in the model above. It can be read using YamlBeans if you are using Java. SQL can be easily generated using Ruby’s (or Python’s) YAML library.

In addition to test data this file also contains data for test cases as described next.

# Test Cases

There are a number of test cases in the yaml file that describe test data and expected results.

|  |  |
| --- | --- |
| **Test Case** | **Description** |
| Test-queues | For each user, test that only tasks for user’s roles are returned and none missing |
| Test-permitted | For each user and permission, test that permitted is correct |
| Test-permitted-users | For permission, test that all permitted users are returned |
| Test-users-for-role | For role and organization, test assigned users are correct and none missing |
| Test-actions | For user and object, check all actions are correct and none missing |

# Test Plan

1. Pick a security framework like Shiro
2. Implement a custom realm to use our tables and take special care to take organization into account when processing permissions
3. Populate tables with test data from permissions.yaml
4. Run test cases as documented in permissions.yaml

# Further Reading

## YamlBeans

<http://yamlbeans.sourceforge.net/>

## Shiro and Spring Security (slightly biased selection)

<http://shiro.apache.org/10-minute-tutorial.html>

<http://stackoverflow.com/questions/4991084/shiro-vs-springsecurity>