Generalizing B-Fabric towards an Infrastructure for Collaborative Research in Switzerland

Deliverable No D3:

Specification of a role-based access model for B-Fabric

March 2010

Document Information

Project	
Project Title	Generalizing B-Fabric towards an Infrastructure for Collaborative Research in Switzerland
Project Start	01.06.2009
Project Sponsor	SWITCH (AAA/SWITCH Project)
Project Number	UZH.5

Document	
Title	Specification of a role-based access model for B-Fabric
Date	
Author(s)	Fuat Akal, Can Türker
Total Number of Pages	9
File Name	D3.docx
Key words	Access model, user roles, role groups

Table of Contents

Summary

Data access control is crucial in Life Sciences data management. B-Fabric controls access to all data on the basis of projects. Only users with the corresponding permissions can access and manipulate data. The current implementation is tightly integrated in the Java application code, and thus is not flexible enough for evolving access roles of users. With this document, we specify a role-based access model for B-Fabric to eliminate this restriction.

1 Role-Based Access Model

Our envisioned role-based access model is simple and straight-forward. It foresees a class named role holding information about a role. Users can be associated with a role by creating a relationship between the user (referred as *Scientist*) and the role classes. To support the composition of role as set of roles, a role can be associated with other roles, too. This is done using a hierarchical relationship – named rolegroup – which links two or more roles together. Figure 1 graphically depicts the main entities of the role-based model together with their relationships.

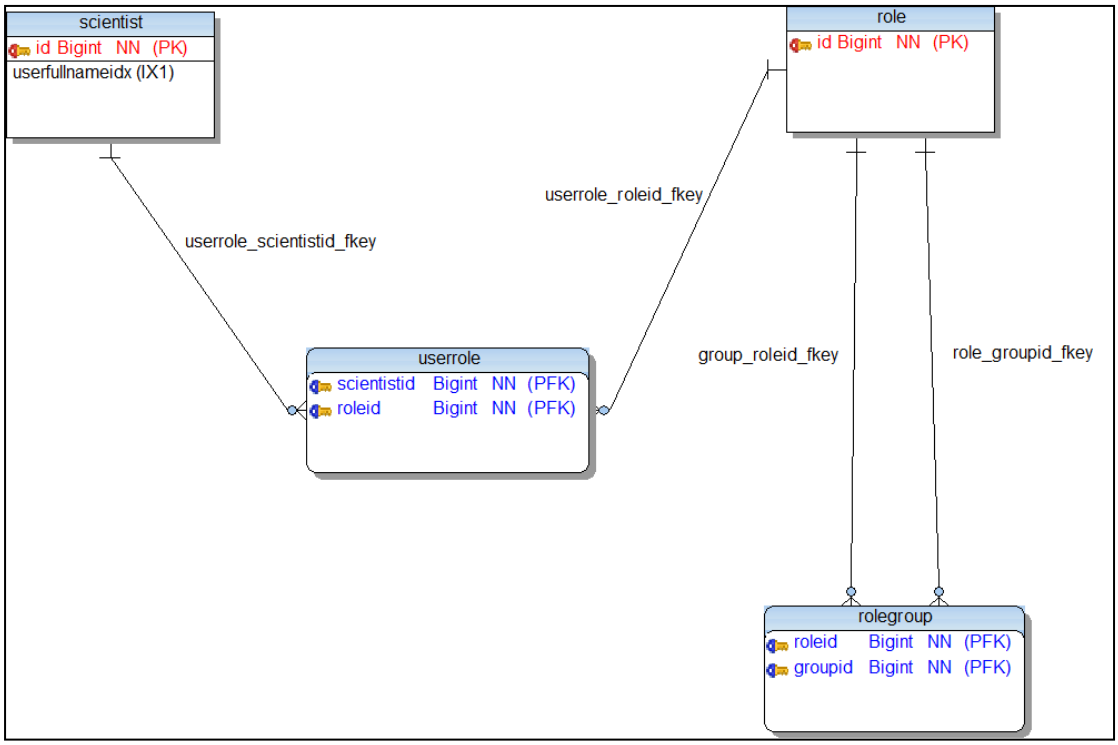


Figure 1: Entities and Relationships for Role-based Access

Entities:

- Scientist: This already existing class contains all relevant information of a B-Fabric user, e.g. id, login, first and last name, address, etc.
- Role: The new class contains all relevant information of a B-Fabric role, i.e., the id and name of a role.
- Rolegroup: This new relationship associates a role with a role group. Several roles can be attached to the same role group.

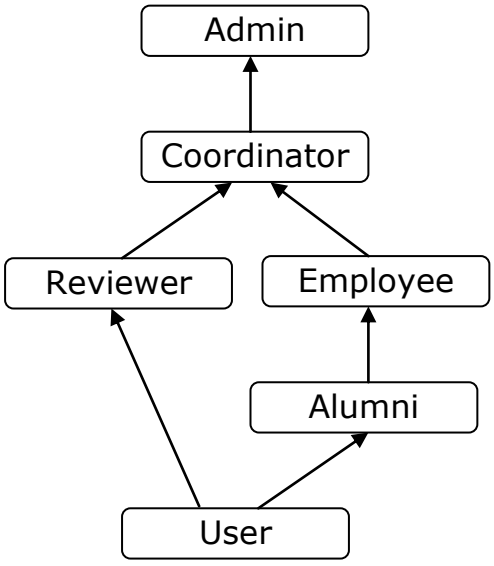
The current FGCZ access strategy foresees the following roles:

- User
- Alumni
- Employee

- Reviewer
- Coordinator
- Admin

Currently, a B-Fabric user has the right to read all and write some data within the projects the user is member of. For instance, the project manager is allowed to write more data than a usual project member, e.g., a project manager can add members to the project or edit certain attributes of the project object.

The above mentioned roles are set into the following role group hierarchy:



Thus, the coordinator role comprises the roles of an employee and a reviewer. A reviewer must not necessarily an employee and vice versa.

2 Tasks

The introduction of the sketched role-based access model implies the following tasks:

1. Extension and adaptation of the current database schema to hold data about roles and relationships between users and roles
2. Migration and adaptation of the current database to this new schema
3. Implementation and adaptation of Java classes to model roles and their relationships
4. Adaptation and revision of the B-Fabric code everywhere permissions are checked

5. Implementation and adaptation of all screens for providing the following functionality at the B-Fabric interface:
   - *Create a new role*
   - *Show the details of a role*
   - *Edit a role*
   - *Delete a role*
   - *Grant/Revoke a role to/from a user*
   - *Grant/Revoke a role to/from a role group*
   - *Show roles of a user*

3 API Definitions

public void createRole(String role) throws Exception

```
@param role Name of the role as a string of maximum 256
characters long.
```

Creates a role in the system. Throws RoleExistsException when the role already exists in the system.

public void deleteRole(String role) throws Exception

```
@param role Name of the role
```

Deletes a role. Throws RoleDoesNotExistException when the role does not exist in the system. Throws RoleInUseException when the role is owned by any user or role.

public grantRoleToUser(String user, String role) throws Exception

```
@param user Login name for the user
@param role Name of the role
```

Grants a role to a user. Throws UserNotFound exception if the user does not exist. Throws RoleNotFound exception if the role does not exist.

public grantRoleToUser(User user, String role) throws Exception

```
@param user Java object that represents the logged in
user
@param role Name of the role
```

Grants a role to a user. Throws UserNotFound exception if the user does not exist. Throws RoleNotFound exception if the role does not exist.

public revokeRoleFromUser(String user, String role) throws Exception

```
@param user Login name for the user
@param role Name of the role
```

Revokes a role from a user. Throws UserNotFound exception if the user does not exist. Throws RoleNotFound exception if the user does not have the role or the role does not exist.

public revokeRoleFromUser(User user, String role) throws Exception

```
@param user Java object that represents the logged in user
@param role Name of the role
```

Revokes a role from a user. Throws UserNotFound exception if the user does not exist. Throws RoleNotFound exception if the user does not have the role or the role does not exist.

public grantRoleToRoleGroup(String roleGroup, String role) throws Exception

```
@param roleGroup Name of the role group
@param role Name of the role
```

Grants a role to a role group. Throws RoleGroupNotFound exception if the role group does not exist. Throws RoleNotFound exception if the role does not exist.

public revokeRoleFromRoleGroup(String roleGroup, String role) throws Exception

```
@param roleGroup Name of the role group
@param role Name of the role
```

Revokes a role from a role group. Throws RoleGroupNotFound exception if the role group does not exist. Throws RoleNotFound exception if the role group does not have the role or the role does not exist.

4 Conclusion

With this delivery we specified a role-based access model for B-Fabric. The envisioned access model supports a straight-forward implementation of roles which can be assigned to users as well as to other roles. We sketched how the current B-Fabric implementation has to be adapted and extended to support this envisioned access model.