The case for Database Query Roles

Joop Ringelberg 11-09-20 Version: 1

Introduction

One of the design goals for Perspectives is that all context- and role instances must be *reachable*. This can be attained by direct indexing (e.g. a role is directly linked to its context), alternatively by deploying an indexed name[1], or finally by a database query that retrieves instances of a particular type.

Such a query has to be the expression by which we define a Calculated Role. To differentiate such database-query-based Calculated Roles from those that are defined by a path query, we call them **Database Query Roles** (DBQ Roles).

Database Query Roles must be context roles: their extension consists of external roles, either all instances of the type, or a filtered subset of them. The extension of a DBQ Role may contain otherwise **unlinked** roles.

Why should we have Database Query Roles?

In this text I describe use cases in the abstract and explore some of their characteristics.

Database Query Roles retrieve context type instances

This is the essence of a Database Query Role: a role defined as such in a context has as its extension all instances of a particular Context type[2] (possibly filtered). A use case might be:

- All natural persons known to a user through Perspectives (through a calculated query that retrieves `User` instances from their `PerspectivesSystem` instance);

In general, it seems that these role types might be used to avoid direct indexing if such indexing would be unpractical:

- Because there would be very many roles in one context, slowing down the system when it context must be loaded, even if just a single role would be accessed;
- Because it would be tiresome to retrieve all instances through a complex path query. This typically happens if the primary structure of contexts has many branches and the wish to have a single list of instances is only of secondary importance.

[1] An indexed name has a different extension (reference value) for each end user, e.g. My System.
[2] In fact, their external roles. In this text I will loosely equate a context with its external role, both on the instance and the type level.

How Database Query Roles are different

To prevent misunderstanding: these roles need no special handling in terms of creating or deleting their instances. The normal assignment and create syntax applies. Furthermore, these roles can be part of a path query.

The operational semantics of deleting is different

We stipulate that an external role instance and its context are removed permanently from the end users bubble[3] whenever

1. It's last binding is removed, or:
2. It is being removed (by user or bot) from a Database Query Role while it turns out to have no bindings.

The consequence of this is that when (1) occurs, no DBQ Role based on the same type will show the instance any more.

The extensional semantics of a DBQ Role

The extension of a DBQ Role is the set of role instances of a particular type (that comply with some optional filter criteria) that

1. either are bound in some role,
2. or have been added directly to some DBQ based on the same type.

Unexpected behaviour

As a consequence of its extensional semantics, these roles show some unusual behaviour. Consider the case of two different contexts (of the same type or of different types, but with a similarly defined Database Query Role). A sub context constructed in either context will be reachable through both. This is unusual behaviour: by default a sub context is only reachable through its containing context.

Still, there is no 'privacy leak'. One might assume such a leak occurs in the following situation: a context A defines three user roles. Two of them have a perspective on a Chat role in the same context, the third role is excluded. The Chat role is defined as a Database Query Role. Now, in another type of context, B, a similarly defined Database Query Role exists and the end user that was excluded in A has a perspective on it in B. Will he be able to see, in B, the Chats that were hidden from him in A?

The answer is no. Even though this user will see (in B) all Chats that are represented in his system, *the other users never sent him their private Chats in A in the first place!* So privacy as modelled is preserved.

[3] The part of the Perspectives Data Universe that is accessible to him.