## Общие положения

Для реализации вхождения группы в группу необходимо исходить от следующих условий:

* ресурсоемкость запросов на получение доменных объектов доступных на чтение должна возрасти незначительно;
* сложность запросов на получение доменных объектов доступных на чтение не должна зависеть от уровня вложенности групп друг в друга;
* не должно быть разницы для включения статических или динамических групп друг в друга

Для реализации данных требований целесообразно хранить развернутую структуру вхождения групп в группы в дополнение к иерархической структурой и обновлять развернутую структуру при изменениях, вносимых в иерархическую структуру. Под развернутой структурой подразумевается сущность, в которой хранится вхождение групп в группы с учетом иерархии. При создание новой группы в сущности хранящей развернутую структуру создается первая запись для новой группы, со ссылкой на саму себя. Это позволяет не менять запросы на получение прав доступа в зависимости от наличия или отсутствия дочерних групп. При получение прав доступа используется только развернутая структура.

## Техническое решение

Создаются две сущности:

1. GROUP\_GROUP\_SETTINGS – конфигурация вхождения групп в группы. Хранятся ссылки только на непосредственно включенные группы. Используется при администрирование и получение непосредственного состава группы. Изменяется с помощью Public API. Доступно для изменения администраторам.

Поля:

PARENT\_GROUP\_ID – родительская группа, ссылка на сущность USER\_GROUP.

CHILD\_GROUP\_ID – дочерняя группа, ссылка на сущность USER\_GROUP.

1. GROUP\_GROUP – развернутая структура вхождения групп в группы. Хранятся ссылки на саму себя, на непосредственно включенные группы и на группы включенные в дочерние группы. Используется при получение всех дочерних групп по родителю. Заполняется по событиям изменения сущности GROUP\_GROUP\_SETTINGS. Запрещается редактирование с помощью public API, недоступно администраторам для редактирования.

Поля:

PARENT\_GROUP\_ID – родительская группа, ссылка на сущность USER\_GROUP.

CHILD\_GROUP\_ID – дочерняя группа, ссылка на сущность USER\_GROUP.

Упрощенная модель данных отображена на рис. 1.

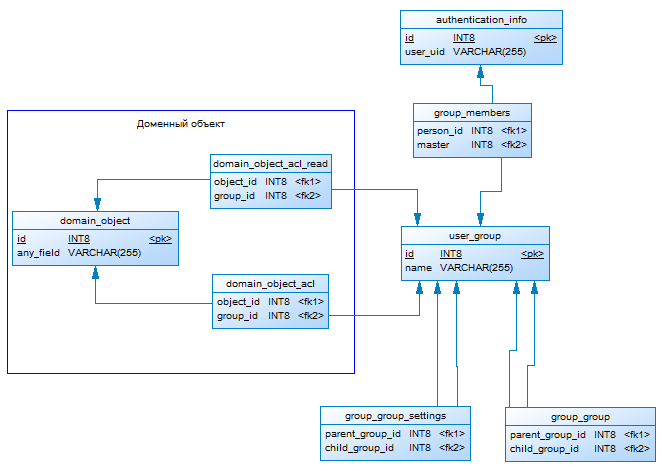


Рис. 1. Упрощенная модель данных, при реализации включения группы в группу.

## Описание алгоритма работы

Необходимо создать сервис для работы с группами (возможно стоит использовать существующий сервис) и метод reclacGroupMembers(groupId), который пересчитывал бы записи в сущности GROUP\_GROUP по идентификатору группы.

При добавление группы в группу необходимо:

* Добавляется запись в GROUP\_GROUP\_SETTINGS
* Для родительской группы и для всех групп куда входит родительская группа в качестве дочерней пересчитываются записи в сущности GROUP\_GROUP вызовом метода reclacGroupMembers

При удаление дочерней группы из родительской следует:

* Удалить запись в GROUP\_GROUP\_SETTINGS
* Для родительской группы и для всех групп куда входит родительская группа в качестве дочерней пересчитываются записи в сущности GROUP\_GROUP вызовом метода reclacGroupMembers

При получение актуальных прав пользователя на доменный объект необходимо модифицировать существующий запросы, приведя их к подобному виду:

Получение acl для доменного объекта:

Select acl.\* from domain\_object\_acl acl

Inner join group\_group gg on (acl.group\_id = gg.parent\_group\_id)

Inner join group\_members gm on (gg.child\_group\_id = gm.master)

Where gm.person\_id = :user\_id

And acl.object\_id = :object\_id

, где

user\_id – идентификатор пользователя из сущности authentication\_info

object\_id – идентификатор доменного обьекта

Получение списка доменных объектов:

Select domain\_object.\* from domain\_object d

Where exists (select acl.object\_id from domain\_object\_acl\_read acl

Inner join group\_group gg on (acl.group\_id = gg.parent\_group\_id)

Inner join group\_members gm on (gg.child\_group\_id = gm.master)

Where gm.person\_id = :user\_id

And acl.object\_id = d.id

)

, где

user\_id – идентификатор пользователя из сущности authentication\_info