Права доступа

Подсистема разграничения доступа (ПРД) CM4/5 управляет разрешениями на выполнение операций пользователями системы. Операции, контролируемые ПРД, можно разделить на 2 класса: контекстные (выполняемые над определёнными объектами) и глобальные (без контекста). Разрешения для контекстных операций могут зависеть от содержания самого объекта доступа (контекста). Контекстом для ПРД могут быть только доменные объекты.

Правила выдачи разрешений (матрица доступа) определяются в конфигурации системы. Для глобальных операций всё просто: правила определяют группы пользователей, которые получат разрешение на выполнение данной операции. О группах речь пойдёт ниже, пока только необходимо подчеркнуть, что разрешения выдаются не отдельным пользователям, а группам.

Для контекстных операций вводится понятие контекстной роли. Под ней мы понимаем набор групп (опять групп!) пользователей, имеющих отношение к данному объекту. При этом отношение может быть совершенно любым, способ его определения отдаётся на откуп настройщику или разработчику: из содержимого полей данного доменного объекта или любых связанных с ним, с помощью SQL-запроса или даже собственного Java-класса. Как бы это ни происходило, контекстная роль, получая параметром доменный объект (контекст), раскрывается в набор групп, которым могут быть предоставлены права на выполнение той или иной операции над этим объектом.

Теперь, наконец, о группах пользователей. Они бывают двух видов: статические и динамические. Состав статических групп определяет администратор через свой интерфейс: он может добавить пользователя в группу или удалить из неё. (Следует отметить, что он не может создавать или удалять сами группы, так как они используются в правилах доступа, определяемых в конфигурации.) Состав динамических групп вычисляется системой. И вновь у настройщика/разработчика имеется большой простор в выборе способа вычисления: SQL-запрос, Java-класс. Но самое интересное в динамических группах то, что они также могут быть контекстными, т.е. зависеть от доменного объекта! Фактически, при определении в конфигурации одной контекстной динамической группы пользователей ПРД создаёт множество реальных групп – столько, сколько есть подходящих доменных объектов. И состав каждой вычисляется исходя из содержимого своего контекстного объекта, т.е. можно снова использовать его поля или связанные объекты.

Важно отметить, что разрешения на конкретные операции выдаются конкретным группам пользователей. И алгоритм, вычисляющий состав контекстной роли, должен определять для контекстной динамической группы её реальный экземпляр. Чаще всего, именно те доменные объекты, на которые ссылается контекстный или связанный с ним объект, будут, в свою очередь, становиться контекстом для динамических контекстных групп пользователей. Правила доступа для глобальных операций могут ссылаться на контекстные группы пользователей, но тоже должны указывать какой-либо метод определения контекста для такой группы.

## Примеры

Примеры статических групп: Администраторы, Делопроизводители. Живой человек (администратор) при необходимости вручную (через административный интерфейс) добавляет и удаляет пользователей в эти группы. (См. также «О проблеме курицы и яйца»).

Примеры динамических неконтекстных групп:

* Все пользователи – система автоматически добавляет туда каждого пользователя (доменный объект «Пользователь»), заведённого в системе.
* Делопроизводители – альтернативный вариант реализации группы, добавляющий пользователя в неё на основании, например, некоторого флага (поля) в его профиле (доменном объекте).

Примеры динамических контекстных групп:

* Пользователь и его заместители – по одной группе на каждый доменный объект «Пользователь», в которую, помимо самого этого пользователя, добавляются также временно или постоянно замещающие его пользователи на основе доменных объектов «Замещение». Алгоритм вычисления группы может учитывать и увольнение сотрудника, удаляя его самого из его же группы, при этом его заместители смогут выполнять его функции.
* Сотрудники подразделения – по одной группе на каждый доменный объект «Подразделение», вычисляемой на основании ссылок на подразделение в доменных объектах «Сотрудник».
* Сотрудники подразделения пользователя – вычисляется аналогично предыдущей, но по одной на каждый доменный объект «Сотрудник».

Примеры контекстных ролей:

* Автор документа (для доменного объекта «Документ») – включает группу «Пользователь и его заместители», контекстом для которой является пользователь из поля «Автор».
* Исполнители поручений (для доменного объекта «Документ») – включает экземпляры группы «Пользователь и его заместители» для всех пользователей, содержащихся в полях «Исполнитель» каждого дочернего объекта «Поручение».
* Подписант документа-основания (для доменного объекта «Поручение») – включает группу «Пользователь и его заместители» с контекстом пользователя из поля «Подписант» дочернего объекта «Подпись» родительского объекта (типа «Документ»).

## Об операциях

Операции, являющиеся предметом назначения прав доступа:

* в контексте доменного объекта с точностью до его типа и статуса:
  + чтение
  + изменение (запись)
  + удаление
  + создание дочернего объекта заданного типа
  + выполнение заданного конфигурируемого действия
* глобальные:
  + создание доменного объекта заданного типа
  + выполнение заданного конфигурируемого действия

Из перечисления видно, что большинство операций суть шаблоны операций, параметризуемые конкретными типами объектов или действиями. Права доступа на каждую такую операцию (например, «Создание дочернего объекта „Поручение“», «Создание дочернего объекта „Подпись“» и т.д.) назначаются независимо друг от друга.

Также необходимо отметить, что в списке отсутствует важнейшая контекстная операция «Изменение статуса». Причина заключается в том, что основным механизмом управления жизненным циклом доменного объекта в системе должны быть бизнес-процессы. Они являются (важной) частью бизнес-логики и работают от имени системы. Ручное изменение статуса пользователем (или иным клиентом) системы рассматривается в качестве исключения, и при наличии такой необходимости оно может быть выставлено в качестве конфигурируемого контекстного действия, операция выполнения которого уже является объектом назначения прав доступа. Конечно, при этом подходе операция изменения статуса доменного объекта не может быть вынесена в публичный API системы.

Та же логика может быть применена и к глобальному действию «Создание доменного объекта заданного вида», но отсутствие этой операции в публичном API видится более нелогичным, поэтому она всё же включена в список доступных для назначения прав операций. Тем не менее, это решение таит в себе ряд неудобств. В интерфейсе системы создание недочерних доменных объектов должно быть конфигурируемой функцией (предусмотрено много различных способов их создания в различных местах), для чего весьма удобную концепцию предоставляют конфигурируемые действия. Однако, их выполнение также является операцией, управляемой ПРД, и мы должны либо требовать от настройщика дублировать правила доступа для этих операций, либо предусматривать способ их логического связывания (и автоматического распространения правил с одной на другую).

Наконец, тот же вопрос стоит и в отношении контекстной операции «Назначение (запуск) заданного бизнес-процесса» (для доменного объекта). Если функции управления бизнес-процессами выносятся в публичный API, то они должны быть под управлением ПРД, а список дополнен соответствующими операциями. Или же эти функции генерализуются через интерфейс конфигурируемых действий, и тогда дополнительные операции в ПРД не нужны.

## Разница между контекстными группами и контекстными ролями

Собственно, зачем же в ПРД нужны контекстные роли? Они очень похожи на контекстные группы пользователей. Внешняя разница между ними заключается лишь в том, что контекстные группы включают в себя пользователей, а контекстные роли – группы пользователей (не обязательно контекстные, но в первую очередь именно их). Любые правила можно технически реализовать с использованием только контекстных групп.

Важнейшее различие между этими объектами заключается в их внутренней реализации. Контекстные группы пользователей существуют физически – в виде записей в БД, по одной на каждый контекстный объект для каждой группы. Хранится также состав каждой такой группы – список пользователей, входящих в неё. При каждом изменении контекста (доменного объекта или связанных с ним, влияющих на состав группы) производится повторное вычисление состава этой группы.

Контекстные роли, с другой стороны, существуют только в конфигурации. Их состав не хранится в БД, а вычисляется непосредственно при вычислении списка доступа к доменному объекту.

Этими свойствами групп и ролей обуславливается разница в их использовании. Контекстные группы пользователей предназначены для использования с относительно редко изменяющимися объектами в качестве контекста, однако позволяют задавать относительно сложные правила их формирования. Для СЭД такими объектами являются пользователи и связанные с ними сущности (профили), организационно-штатная структура и т.п. Контекстные роли связываются с динамичными объектами системы (документы, поручения и т.п.), существующими в большом количестве, но должны иметь максимально лёгкие и быстрые алгоритмы для их вычисления.

## Списки доступа

Списки доступа (Access control list, ACL) – внутренний объект ПРД, предназначенный для оптимизации запросов определения доступа.

Списки доступа для контекстных операций хранятся в таблицах БД – по одной таблице на каждый тип доменных объектов. Списки доступа хранят *актуальные* разрешения (permissions) на операции над каждым доменным объектом. Субъектами разрешений выступают группы пользователей. Актуальность означает хранение только тех разрешений, которые действительны для текущего статуса доменного объекта. При изменении статуса объекта производится повторное вычисление списка доступа к нему.

ПРД также отслеживает все изменения доменных объектов и другие события, которые могут привести к изменению списков доступа, и вносит соответствующие изменения в списки.

Наличие списков доступа позволяет реализовать проверку прав пользователя на выполнение той или иной операции (включая операции над множеством однотипных объектов, например, чтение списка) через обращение лишь к двум таблицам БД: списку доступа (частному для этого типа объектов) и составу групп пользователей.

## Доступ к полям объектов

В некоторых случаях выдвигаются требования дополнительного разграничения доступа к отдельным полям объектов (как на чтение, так и на изменение). Непосредственная реализация этих требований может привести к чрезмерному росту как объёма списков доступа, и без того немалого, так и количества вычислений этих списков.

Для удовлетворения подобных требований можно предложить использовать декомпозицию доменных объектов на такие части, что правила доступа ко всем полям одной части будут одинаковы, и соединение этих доменных объектов в иерархию.

## О проблеме курицы и яйца

Уже упоминалось, что управление составом групп пользователей осуществляет администратор. Но сам администратор – это не кто иной, как пользователь системы, включённый в группу «Администраторы». Но откуда же возьмётся самый первый администратор? (Подобная проблема, кстати, встаёт и с пользователями вообще – управление пользователями обычно осуществляет также администратор, но он сам является пользователем, и кто-то должен добавить его самого.)

Для решения этой проблемы при старте системы производится специальная проверка. Если группа «Администраторы» пуста (не содержит ни одного пользователя), создаётся специальный пользователь system\_administrator (если он ещё не существует) и добавляется в группу «Администраторы». Это не только решает проблему первого старта, но и защищает систему от случайного удаления всех администраторов. Конечно, доменные объекты «Пользователь», «Группа пользователей», а также правила, предоставляющие права на изменение этих объектов группе «Администраторы» задаются в системной конфигурации (недоступной для изменения настройщиком).

## Дополнительные функции

С учётом тотального преобладания количества разрешений на чтение над количеством разрешений на любые другие операции над объектом, а также количества запросов прав чтения над всеми прочими, целесообразно рассмотреть возможность разбиения списков доступа на две отдельные таблицы (для каждого типа доменных объектов, конечно): списки чтения и списки доступа (остальные операции).

Требуется реализовать возможность включения одних групп пользователей в другие. При этом включение может быть вложенным (на любую глубину). Такая функция была реализована в прототипе системы. Детали конфигурирования и реализации будут изложены позже.

С учётом наличия в системе большого количества служебных доменных объектов (в частности, для реализации полей с множественными значениями) необходимо реализовать возможность «наследования» списков доступа дочерними объектами от родительских. Т.е. для конкретного типа доменного объекта в конфигурации может быть указано использование родительских списков доступа; в таком случае вместо собственных таблиц для него создаётся отдельная таблица (или поле) с указателем на имеющего список доступа родителя (т.к. наследование может быть многоуровневым), а запросы определения прав должны использовать списки доступа этого родителя.

В следующих версиях системы планируется дополнительная реализация мандатного принципа доступа. При этом для выполнения операции пользователь должен будет иметь разрешения по обоим принципам: и дискреционному (нынешнему), и мандатному.

# Принципы реализации

1. **Проверка прав (авторизация) пользователя осуществляется** кодом, расположенным **в слое бизнес-логики (сервисов)**. Весьма желательно, чтобы был выделен отдельный компонент, осуществляющий такие проверки (требования безопасности + единый код). Для обеспечения высокой производительности системы возможно делегирование проверок на слой DAO (встраивание в запросы – особенно, при чтении списков объектов), однако факт необходимости проверки и идентификатор пользователя, для которого осуществляется проверка, устанавливается вызывающим кодом (параметр).
2. Поскольку операция получения (чтения) доменных объектов является наиболее часто используемой (один запрос пользователя может приводить к десяткам и даже сотням операций чтения доменных объектов), а асинхронное выполнение таких операций невозможно, **скорость выполнения проверок при чтении доменных объектов должна быть максимально высокой**. Скорость выполнения проверок прав выполнения других операций тоже важна, но не столь критична.
3. Главными событиями, влияющими на состав списков доступа, являются изменения доменных объектов. При этом изменение единичного доменного объекта может приводить к изменениям списков доступа для большого количества других (связанных) объектов. **Очень важно обеспечить высокую скорость обновления списков доступа**, поскольку выполнение этих операций планируется производить в синхронном режиме.
4. Нередко в пределах одной транзакции выполняется модификация нескольких объектов, как правило, связанных. При этом весьма вероятно воздействие этих изменений на списки доступа одних и тех же объектов. Также возможен вариант многократных изменений одного и того же объекта в рамках одной транзакции. Всё это позволяет оптимизировать операции пересчёта списков доступа, **накапливая в течение транзакции список объектов**, для которых необходим пересчёт, и **выполняя** этот **пересчёт перед завершением транзакции**.
5. Изменение конфигурации системы в части изменения правил доступа и контекстных ролей может приводить к необходимости **пересчёта колоссального количества списков доступа** (миллионы доменных объектов). Необходимо предусмотреть меры, обеспечивающее **разумное время выполнения** этих операций (не более десятков минут) и защиту от ошибок.
6. Помимо контекстных (проводимых над объектами) операций система предусматривает возможность выполнения глобальных операций (например, создание доменного объекта). Права на такие операции, конечно, не могут зависеть от контекстных ролей. Они зависят лишь от правил доступа (конфигурация, которая хранится в оперативной памяти) и членства пользователя в группах. Представляется разумным не хранить списки доступа к таким операциям в БД, а **вычислять возможность выполнения пользователем конкретной операции по запросу**.
7. Сервис управления доменными объектами (CrudService) должен иметь **методы**, позволяющие выполнить **проверку возможности выполнения** той или иной **операции**. Из соображений производительности имеет смысл также добавить метод, возвращающий все позволенные пользователю операции над конкретным объектом.
8. Таблицы со списками доступа к объектам создаются одновременно с таблицами, хранящими сами эти объекты (при изменении конфигурации).
9. **Код операции**, хранящийся в списках операций (таблицы \*\_ACL), **является внутренней информацией слоя хранения** и не должен выходить за его пределы. Интерфейс CrudService должен обеспечивать его преобразование в классы или enum. Особенно важно это для составных кодов (C\_*type*, E\_*action*) , представление которых позволит клиенту явно получить имя типа или действия.
10. В дальнейшем (не в первую очередь) потребуется создание компонента, который сможет «разъяснять» пользователям состав списка доступа к любому объекту, т.е. выводить информацию о том, на каком основании (по какому правилу) было предоставлено право на конкретную операцию с объектом тому или иному пользователю.