МАТЕМАТИЧЕСКИЕ ОСНОВЫ КОМПЬЮТЕРНОЙ БЕЗОПАСНОСТИ

DOI 10.17223/20710410/21/6

УДК 004.94

ОСНОВНЫЕ ЭЛЕМЕНТЫ МАНДАТНОЙ СУЩНОСТНО-РОЛЕВОЙ ДП-МОДЕЛИ УПРАВЛЕНИЯ ДОСТУПОМ И ИНФОРМАЦИОННЫМИ ПОТОКАМИ В СУБД POSTGRESQL ОС СПЕЦИАЛЬНОГО НАЗНАЧЕНИЯ ASTRA LINUX SPECIAL EDITION

А. В. Шумилин

ОАО «НПО РусБИТех», г. Москва, Россия

E-mail: a.shumilin@rusbitech.ru

Рассмотрен подход к разработке на основе мандатной сущностно-ролевой ДП-модели управления доступом и информационными потоками в операционной системе (ОС) семейства Linux (МРОСЛ ДП-модели) новой ДП-модели, адаптированной для реализации в СУБД PostgreSQL, применяемой в ОС специального назначения Astra Linux Special Edition (PostgreSQL ДП-модели). Описываются её новые элементы, отличия от МРОСЛ ДП-модели и от известных автору моделей управления доступом в СУБД, в том числе учитывающие особенности управления доступом к данным, хранящимся и обрабатываемым в исследуемой СУБД.

Ключевые слова: *компьютерная безопасность, системы управления базами данных, ролевая ДП-модель, Astra Linux.*

1. Введение. Необходимость адаптации модели. Этапы разработки

При разработке автоматизированных систем в защищённом исполнении необходимо обеспечивать соответствие требованиям по защите информации от несанкционированного доступа (НСД). Эти требования зависят от конфиденциальности информации и характера доступа к ней пользователей и приведены в руководящих документах ФСТЭК России [1, 2]. Выполнение требований должно, в том числе, подтверждаться сертификатом соответствующей системы сертификации на программное обеспечение. Для обеспечения выполнений указанных требований разработчики защищённых программных комплексов (ЗПК) применяют те или иные из существующих классических моделей управления доступом зачастую без учёта всех условий и особенностей функционирования современных программно-аппаратных средств. При этом больше внимания уделяется созданию непосредственно механизмов защиты, чем построению непротиворечивых формальных моделей логического управления доступом в разрабатываемых системах. Применение моделей, позволяющих провести анализ этих механизмов безопасности, при условии обеспечения гарантий проектирования и реализации, повышает доверие к безопасности создаваемых ЗПК. Следует отметить, что в современных условиях уже недостаточно обеспечения управления доступом в соответствии с дискреционными и мандатными моделями управления доступом без учёта условий возникновения запрещённых информационных потоков.

В настоящее время осуществляются попытки построения современных формальных моделей, объединяющих все аспекты дискреционного, мандатного и ролевого

управления доступом с учётом безопасности информационных потоков, для применения в реальных защищённых ОС. Например, при создании перспективной операционной системы специального назначения (ОС СН) Astra Linux Special Edition используется мандатная сущностно-ролевая ДП-модель управления доступом и информационными потоками в операционных системах семейства Linux [3 – 5].

Поскольку неотъемлемой частью системы обработки данных в автоматизированных системах являются системы управления базами данных (СУБД), они также должны отвечать требованиям по защите информации от НСД, применяемым к создаваемой системе в целом. При построении автоматизированных систем в защищённом исполнении, как правило, используются СУБД, функционирующие под управлением применяемых ОС. В этом случае для реализации механизмов защиты в СУБД руководствуются моделью управления доступом, используемой в ОС.

Таким образом, в условиях постоянно возрастающих требований к обеспечению защиты информации при её хранении и обработке реализация механизмов управления доступом в СУБД не только не теряет свою актуальность, но и требует новых подходов, согласующихся с теми моделями управления доступом, которые реализуются в среде функционирования СУБД, которой является ОС.

В настоящее время для обеспечения единого подхода к управлению доступом в СУБД и ОС СН Astra Linux Special Edition автором осуществляется адаптация МРОСЛ ДП-модели с учетом специфики реляционных баз данных. В ходе работы использовались как классические базовые модели управления доступом семейства RBAC [6, 7], так и современные результаты теоретического моделирования управления доступом в СУБД [8, 9]. Кроме того, учитывались подходы к реализации управления доступом в СУБД Oracle, SQL Server и PostgreSQL, базирующиеся на требованиях стандарта SQL [10 – 13].

Целесообразность адаптации МРОСЛ ДП-модели и построения на её основе новой PostgreSQL ДП-модели связана со следующими особенностями управления доступом к данным, хранящимся и обрабатываемым в СУБД:

— первичная субъект-сессия СУБД создается субъект-сессией ОС;
— в качестве множества сущностей рассматриваются объекты реляционных баз данных;
— доступ к данным осуществляется с помощью языка запросов SQL;
— множество видов доступа, прав доступа и правила распространения прав доступа регламентируются стандартом SQL;
— в СУБД содержится не только пользовательская, но и служебная информация, описывающая структуру пользовательских данных.

Применение специальных моделей управления доступом в СУБД наряду с реализацией выполнения заданных требований по защите информации должно обеспечивать без потери производительности сохранение неотъемлемых качеств СУБД, таких, как:

— целостность и согласованность данных;
— управление данными и их структурой с помощью языка запросов;
— резервное копирование и восстановление.

Для обеспечения перечисленных качеств язык запросов должен быть расширен соответствующими командами санкционированного изменения правил разграничения доступа (ПРД), позволяющими не только гибко управлять ими, но и поддерживать резервное копирование и восстановление баз данных без потери информации о ПРД.

Создание эффективных механизмов управления доступом в реляционных СУБД требует не только хорошего знания реализуемых формальных моделей, но и детального анализа архитектуры и особенностей функционирования реальных систем обработки данных [14 – 17].

В связи со сложностью решение поставленной задачи выполняется поэтапно.

На первом этапе произведён анализ существующих механизмов управления доступом в рассматриваемой СУБД и возможности использования элементов МРОСЛ ДП-модели.

На втором этапе строится PostgreSQL ДП-модель, при этом формулируются определения и положения, необходимые для описания специфики СУБД ОС СН, разрабатываются правила преобразования состояния системы в рамках модели для нескольких существенных операций SQL. Для оценки возможности использования модели на её основе реализуется механизм управления доступом в реальной СУБД, что требует решения многих практических задач [18].

На третьем этапе предполагается расширение PostgreSQL ДП-модели правилами преобразования для большинства команд SQL, формулирование и обоснование в рамках модели, как минимум, достаточных условий безопасности механизма управления доступом и информационными потоками в рассматриваемых СУБД и ОС СН.

2. Элементы модели (описание состояния системы)

При разработке PostgreSQL ДП-модели были взяты за основу элементы МРОСЛ ДП-модели, задающие мандатное и ролевое управление доступом и информационными потоками. При этом для упрощения не рассматривались системы ограничений ролевого управления доступом и мандатный контроль целостности, которому соответствуют некоторые механизмы современных ОС (MIC — Mandatory Integrity Control и UAC — User Account Control в ОС семейства Microsoft Windows).

В отличие от МРОСЛ ДП-модели, для ОС в разрабатываемой модели роли трактуются не как сущности-объекты файловой системы, а как сущности-объекты СУБД. Множество ролей СУБД не является подмножеством ролей ОС, то же относится и к множеству административных ролей. Особые административные роли МРОСЛ ДП-модели для ОС не являются ролями СУБД и не могут быть получены пользователем СУБД непосредственно. Для обеспечения поддержки этих административных функций в PostgreSQL ДП-модели вводятся соответствующие параметры субъект-сессии СУБД, которые наследуются при создании субъект-сессии СУБД от субъект-сессии ОС.

В качестве множества сущностей (объектов и контейнеров) с заданной на нём иерархической структурой рассматриваются носящие подобный характер объекты реляционных баз данных (БД), применяемые в СУБД PostgreSQL. Следует отметить, что поскольку записи базы данных содержат в своём составе мандатный уровень конфиденциальности, то они рассматриваются в модели в качестве объектов, а содержащие их таблицы — соответственно в качестве контейнеров.

Системный каталог (метаданные) рассматривается как самостоятельная БД, реализованная с помощью средств СУБД. При этом все операции с этой БД осуществляются с помощью специальных конструкций языка запросов SQL или привилегированным пользователем в специальном режиме. В модели рассматривается управление доступом к пользовательским данным. Доступ к таблицам, записям таблиц и остальным объектам системного каталога осуществляется аналогично, за исключением того, что они считаются принадлежащими привилегированному пользователю.

В отличие от файловой системы ОС, иерархия объектов в СУБД носит более детерминированный характер, так как имеет ограниченную степень вложенности разнотипных объектов (например, база данных/схема/таблица/запись). Это позволяет привнести в модель элементы типизированной матрицы доступа, так как каждая операция SQL содержит информацию о типе объекта. Таким образом, PostgreSQL ДП-модель была расширена соответствующими определениями иерархии типов сущностей. Используем следующие обозначения (аналогичные применяемым в исходной МРОСЛ ДП-модели и работах [3, 8, 9]):

- $E = O \cup C$ — множество сущностей, где O — множество объектов, а C — множество контейнеров и $O \cap C = \varnothing$;
- S — множество субъект-сессий пользователей;
- T — множество типов сущностей;
- $type : E \to T$ — функция типов сущностей (по аналогии с [8]);
- R — множество ролей;
- AR — множество административных ролей, при этом по определению $AR \cap R = \varnothing$;
- $H_R : R \cup AR \to 2^R \cup 2^{AR}$ — функция иерархии ролей и административных ролей;
- U — множество учётных записей пользователей, при этом учётные записи в PostgreSQL являются подмножеством ролей, т. е. $U \subseteq R \cup AR$;
- $user : S \to U$ — функция принадлежности субъект-сессии учётной записи пользователя, задающая для каждой субъект-сессии учётную запись пользователя, от имени которой она активизирована;
- $R_a = \{read_a, write_a, append_a\}$ — множество видов доступа;
- $R_f = \{write_m, write_t\}$ — множество видов информационных потоков (по памяти и по времени);
- R_r — множество видов прав доступа, элементы которого с учётом специфики рассматриваемой СУБД будут определены далее;
- $R_{raf} = R_r \cup R_a \cup R_f$ — множество видов прав доступа, видов доступа и видов информационных потоков, при этом множества R_r, R_a и R_f попарно не пересекаются;
- $P \subseteq E \times R_r$ — множество прав доступа к сущностям;
- $PA : R \cup AR \to 2^P$ — функция прав доступа к сущностям ролей и административных ролей;
- $F \subseteq (E \cup R \cup AR) \times (E \cup R \cup AR) \times R_f$ — множество информационных потоков;
- $A \subseteq S \times E \times R_a$ — множество доступов субъект-сессий к сущностям;
- $AA \subseteq S \times (R \cup AR) \times R_a$ — множество доступов субъект-сессий к ролям или административным ролям;
- L_U — множество учётных записей доверенных пользователей;
- N_U — множество учётных записей недоверенных пользователей, при этом по определению справедливы равенства $L_U \cup N_U = U$, $L_U \cap N_U = \varnothing$;
- L_S — множество доверенных субъект-сессий;
- N_S — множество недоверенных субъект-сессий, при этом по определению справедливо равенство $L_S \cap N_S = \varnothing$.

Сохранены все определения исходной модели, касающиеся мандатного контроля конфиденциальности:

- $(LC, \leqslant)$ — решётка многоуровневой безопасности уровней конфиденциальности;

— $(f_u, f_e, f_r, f_s) \in FC$ — четвёрка функций уровней конфиденциальности:
  — $f_u : U \to LC$ — функция, задающая для каждой учётной записи пользователя её уровень доступа — максимальный разрешённый уровень доступа субъект-сессий, функционирующих от её имени;
  — $f_e : E \to LC$ — функция, задающая уровень конфиденциальности для каждой сущности;
  — $f_r : R \cup AR \to LC$ — функция, задающая для каждой роли или административной роли её уровень конфиденциальности;
  — $f_s : S \to LC$ — функция, задающая для каждой субъект-сессии её текущий уровень доступа;
— $CCR : E \cup R \cup AR \to \{\texttt{true}, \texttt{false}\}$ — функция, задающая способ доступа к сущностям внутри контейнеров или ролям в иерархии ролей (с учётом их мандатных уровней конфиденциальности).

Для учёта специфики PostgreSQL дополнительно детализируем состав сущностей, их типов, а также соответствующие функции иерархии:

— $O = O_d \cup O_{col} \cup O_{proc} \cup O_{trg} \cup O_{cnstr} \cup O_{typ} \cup O_{seq} \cup O_{blob} \cup O_{rule} \cup O_{spc} \cup O_{glob}$,
где O_d — множество сущностей данных, соответствующих записям таблиц; O_{col} — множество столбцов; O_{proc} — множество хранимых процедур; O_{trg} — множество триггеров; O_{cnstr} — множество ограничений целостности; O_{typ} — множество используемых в СУБД типов данных; O_{seq} — множество последовательностей; O_{blob} — множество больших двоичных объектов; O_{rule} — множество правил преобразования (*RULE*); O_{spc} — множество табличных пространств (*TABLESPACE*); O_{glob} — множество глобальных объектов (*CAST*, *EXTENSION*, *FOREIGN DATA WRAPPER*, *FOREIGN SERVER*, *LANGUAGE*);
— $coltype : O_{col} \to O_{typ}$ — функция типов столбцов, сопоставляющая каждому столбцу его тип данных (домен);
— $C = C_{cl} \cup C_{db} \cup C_{nsp} \cup C_{rel}$,
где C_{db} — множество контейнеров, соответствующих базам данных; C_{nsp} — множество схем; C_{rel} — множество отношений, включающих таблицы и представления (далее — таблицы); C_{cl} — множество кластеров СУБД PostgreSQL. Кластер является в модели корневым контейнером, в котором располагаются глобальные объекты, такие, как базы данных, табличные пространства, глобальные описания и роли (субъекты);
— $T = \{cl, spc, db, nsp, rel, col, proc, trg, cnstr, typ, seq, blob, rule, glob, d\}$ — множество типов сущностей СУБД (соотносятся с одноимёнными индексами сущностей);
— $H_T : T \to 2^T$ — функция иерархии типов сущностей СУБД (сопоставляющая каждому типу сущности множество типов сущностей $H_T(t) \subset T$, которые в ней могут содержаться согласно особенностям реализации СУБД PostgreSQL), такая, что:
  — если $t \notin \{cl, db, nsp, rel\}$, то $H_T(t) = \varnothing$;
  — если $t = cl$, то $H_T(t) = \{db, spc\}$;
  — если $t = db$, то $H_T(t) = \{nsp, blob, glob\}$;
  — если $t = nsp$, то $H_T(t) = \{rel, proc, type, seq\}$;
  — если $t = rel$, то $H_T(t) = \{d, col, trg, cnstr, type, rule\}$;
— $H_E : E \to 2^E$ — функция иерархии сущностей (сопоставляющая каждой сущности $e \in E$ множество сущностей $H_E(e) \subset E$, непосредственно в ней содержащихся), удовлетворяющая условиям:

- если сущность $e \in H_E(c)$, то $e < c$ и не существует сущности-контейнера $d \in C$, такой, что $e < d$, $d < c$;
- для любых сущностей $e1, e2 \in E$, $e1 < e2$, выполняется $H_E(e1) \cap H_E(e2) = \varnothing$;
- для любых сущностей $e1, e2 \in E$, $e1 < e2$, выполняется $type(e1) \in H_T(type(e2))$ (контроль по типу сущностей);
- если $o \in O$, то справедливо равенство $H_E(o) = \varnothing$.

В исследуемой СУБД применяются дискреционные и ролевые механизмы управления доступом, отличные от механизмов, используемых в ОС. Определено больше видов прав доступа, и они могут применяться по-разному для разных типов сущностей. Язык запросов SQL позволяет гибко управлять правами доступа к объектам и к ролям. Но эта гибкость с точки зрения безопасности информационных потоков является недостатком: множество ролей SQL не разделяется по административному признаку, что в сочетании с возможностью владельца сущности предоставлять доступ к ней произвольным ролям приводит к неконтролируемому распространению прав доступа и возможности создания запрещённых информационных потоков. Для устранения этого было предложено ввести в модель (по аналогии с МРОСЛ ДП-моделью) разделение ролей SQL на административные и неадминистративные с введением ограничений на исполнение команд и изменение правил управления доступом для неадминистративных ролей.

Отличие предлагаемой модели заключается в том, что в ней, опираясь на существующие модели ролевого управления доступом, делается попытка сочетать исходную МРОСЛ ДП-модель с требованиями стандарта на язык запросов SQL [12, 13]. Несмотря на введённое разделение ролей на административные и неадминистративные, управление правами доступа и ролями сохраняется согласно реализации PostgreSQL.

В классической модели управления доступом в СУБД присутствует роль привилегированного пользователя (суперпользователя), которому разрешены все операции в СУБД. В СУБД PostgreSQL существуют особая учётная запись postgres, которая носит такой характер, и особая привилегия роли *SUPERUSER* (*CREATEUSER*), предоставляющая роли аналогичные права.

С учётом специфики СУБД PostgreSQL зададим множество видов прав доступа:

- $R_r = \{select_r, update_r, insert_r, delete_r, truncate_r, references_r, trigger_r, create_r, usage_r, connect_r, execute_r, createdb_r, login_r, createrole_r, superuser_r, own_r\}$.

  В отличие от [9], вместо функции *owner* для задания владельца сущности используется принятое в ДП-моделях [7] право доступа владения own_r, так как оно точнее отражает специфику работы механизмов разграничения доступа в СУБД PostgreSQL. При этом выполняются следующие условия:
  - в каждый отдельный момент времени только одна роль может обладать правом владения к конкретной сущности;
  - роль, создавшая сущность, автоматически получает право владения к ней;
  - принадлежащие таблице объекты (доступ к которым явно не управляется) считаются принадлежащими владельцу таблицы, т. е. для $t \in C_{rel}$, $e \in H_E(t)$, $r \in R \cup AR$ выполняется условие: если $(t, own_r) \in PA(r)$, то $(e, own_r) \in PA(r)$;
  - для глобальных объектов, не имеющих владельца (кластер, *CAST* и т. п.) наличие права доступа $superuser_r$ эквивалентно наличию права доступа владения own_r;
- $R_{ck} = \{createdb_r, login_r, createrole_r, superuser_r\}$ — множество видов прав доступа, применимых к кластеру;

— $R_{db} = \{create_r, usage_r, connect_r, own_r\}$ — множество видов прав доступа, применимых к базе данных;
— $R_{spc} = \{create_r, own_r\}$ — множество видов прав доступа, применимых к табличному пространству;
— $R_{nsp} = \{create_r, usage_r, own_r\}$ — множество видов прав доступа, применимых к схеме;
— $R_{rel} = \{select_r, update_r, insert_r, delete_r, truncate_r, references_r, trigger_r, own_r\}$ — множество видов прав доступа, применимых к таблицам;
— $R_{col} = \{select_r, update_r, insert_r, references_r\}$ — множество видов прав доступа, применимых к столбцам таблиц;
— $R_{seq} = \{select_r, update_r, usage_r, own_r\}$ — множество видов прав доступа, применимых к последовательностям;
— $R_{type} = \{usage_r, own_r\}$ — множество видов прав доступа, применимых к типам и доменам;
— $R_{proc} = \{execute_r, own_r\}$ — множество видов прав доступа, применимых к процедурам;
— $R_{blob} = \{select_r, update_r, own_r\}$ — множество видов прав доступа, применимых к большим двоичным объектам;
— $ALL_r : E \to 2^{R_r}$ — функция применимых к сущности видов прав доступа, такая, что для каждой $e \in E$ по определению справедливо $ALL_r(e) = R_\alpha \alpha \in type(e)$.

В СУБД PostgreSQL в каждый момент времени субъект-сессия может исполняться только с одной активной ролью. Активация другой роли (*SET ROLE*) в модели рассматривается как инициация новой субъект-сессии. Согласно стандарту SQL, сессия в случае наличия активной роли исполняется от её имени, в противном случае — от имени роли учётной записи пользователя, её активизировавшего (функция *user*). В отличие от [9], вводится функция *role*:

— $role : S \to R \cup AR$ — функция принадлежности субъект-сессии роли, задающая для каждой субъект-сессии активную роль, с которой она выполняется.

Функция *role* может меняться при переходе системы из одного состояния в другое, так как в зависимости от режима выполнения кода сущности-процедуры возможно применение прав, отличных от прав вызвавшей такую сущность роли.

В СУБД существуют следующие особенности применения доступов субъект-сессии к сущностям как в отношении пользовательских данных, так и в отношении системных объектов (метаданных):

— для ролей рассматриваются только доступы на чтение и запись, так как в СУБД PostgreSQL отсутствует понятие владельца роли, при этом доступом на запись к роли может обладать только доверенная субъект-сессия;
— для сущностей-контейнеров доступ на чтение позволяет получить список содержащихся в них сущностей, на добавление — возможность создать в них новую сущность, на запись — изменить или удалить сущность в их составе;
— операции по модификации свойств сущностей, в том числе состава таблицы (столбцы, ограничения и т. п.), требуют у субъект-сессии наличия доступного по иерархии права доступа владения;
— операции с записями таблиц приводятся к видам доступа общепринятым образом: *SELECT* — чтение, *INSERT* — добавление, *UPDATE*, *DELETE* — запись.

По аналогии с СУБД ДП-моделью [9] используем обозначения:

— $execute\_as : O_{proc} \to \{as\_caller, as\_owner\}$ — функция, задающая режим выполнения кода сущности-процедуры p субъект-сессией s при следующих условиях:
  — если $p \in O_{proc}$ и $execute\_as(p) = as\_caller$, то код исполняется от имени роли $role(s)$;
  — если $p \in O_{proc}$ и $execute\_as(p) = as\_owner$, то код исполняется от имени роли, непосредственно имеющей право доступа владения к p, т. е. роли $r \in R \cup AR$, для которой выполняется $(p, own_r) \in PA(r)$;
— $operations : O_p \to OP^*$ — функция, задающая для каждого объекта-процедуры конечную последовательность де-юре правил преобразования состояний модели, выполняющихся при её активизации, где OP — множество правил преобразования состояний модели;
— $triggers : C_{rel} \times \{update_r, insert_r, delete_r, truncate_r\} \to O^*_{trg}$ — функция, задающая для таблицы конечную последовательность триггеров, выполняющихся при реализации к нему права доступа определённого вида.

Один и тот же триггер в СУБД PostgreSQL может вызываться при реализации любого из заданных при его создании вида права доступа. При этом последовательность вызова триггеров для одной таблицы определяется в алфавитном порядке их имён. С триггером ассоциирована реализующая его функция:

— $trigger\_proc : O_{trg} \to O_{proc}$ — функция, задающая для триггера реализующую его процедуру.

В СУБД PostgreSQL существует возможность задавать для таблицы специальное правило, которое вызывается при реализации любого из заданных при его создании вида права доступа (аналогично триггеру). При этом последовательность вызова правил для одной таблицы определяется в алфавитном порядке их имён. Обозначим:

— $rules : C_{rel} \times \{select_r, update_r, insert_r, delete_r\} \to O^*_{rule}$ — функция, задающая для таблицы конечную последовательность правил, выполняющихся при реализации к ней права доступа определённого вида;
— $rule\_op : O_{rule} \to \{access\_select, access\_insert, access\_update, access\_delete\}$ — функция, задающая для правила таблицы соответствующее правило преобразования состояния модели из набора правил работы с данными в таблице.

Для обеспечения поддержки специальных административных ролей МРОСЛ ДП-модели в PostgreSQL ДП-модели вводятся соответствующие параметры субъект-сессии СУБД, которые наследуются при создании субъект-сессии СУБД от субъект-сессии ОС:

— $sop : S \to \{downgrade\_admin\_sop, \ldots\}$ — функция получения параметров новой субъект-сессии СУБД из субъект-сессии ОС, такая, что при наличии у субъект-сессии ОС доступа на чтение к административным ролям ОС (например, *downgrade_admin_role* в МРОСЛ ДП-модели) она отображает их в соответствующие параметры (например, *downgrade_admin_sop*);
— $session\_options : S \to \{downgrade\_admin\_sop, \ldots\}$ — функция, определяющая для каждой субъект-сессии её параметры.

Для описания процесса делегирования прав доступа аналогично [9] введено множество $Gr \subseteq (R \cup AR) \times P$ — множество прав доступа на сущности, которые могут быть переданы ролью другой роли. Могут быть переданы только права доступа, которыми роль уже обладает к сущностям, которыми владеет, либо полученные к другим сущностям с опцией *WITH_GRANT_OPTION*, т. е. $Gr(r) \subseteq PA(r)$.

Опция *WITH_ GRANT_ OPTION* даёт возможность получающей право роли передавать полученное право другим ролям. Владелец сущности по умолчанию получает все права на неё и может передавать их другим ролям. При отборе прав доступа возможен отбор опции *WITH_ GRANT_ OPTION*. Право владения не может быть делегировано. В СУБД используется ключевое слово *PUBLIC*, обозначающее права доступа всех существующих ролей — аналог прав для всех в ОС. Данный элемент не является сущностью (ролью или группой, хотя и может рассматриваться как встроенная группа, в которую входят все существующие роли и те, что будут созданы в будущем) и соответственно не может получить опцию делегирования прав доступа *WITH_ GRANT_ OPTION*. Для отражения этой особенности СУБД в модели задаётся следующая роль:

— $public \in R$ — особая наименьшая в иерархии роль, используемая для задания общих для всех прав доступа, такая, что
  — для любой $r \in R \cup AR$ выполняется $public \in H_R(r)$;
  — не существует такой роли $r \in R \cup AR$, что $r < public$.

При создании некоторых сущностей ряд видов прав доступа предоставляется всем ролям с помощью *PUBLIC* ($connect_r$ — для баз данных, $execute_r$ — для процедур и $usage_r$ — для некоторых глобальных объектов). Состав прав доступа по умолчанию может быть изменён администратором.

В отличие от МРОСЛ ДП-модели, осуществляется возврат к следующим обозначениям:

— $UA : U \to 2^R$ — функция авторизованных ролей учётной записи пользователя, задающая для каждой учётной записи пользователя множество ролей, на которые она может быть авторизована;
— $AUA : U \to 2^{AR}$ — функция авторизованных административных ролей учётной записи пользователя, задающая для каждой учётной записи пользователя множество административных ролей, на которые она может быть авторизована.

Аналогично передаче прав доступа осуществляется и управление иерархией ролей. Для описания этого процесса дополнительно введено множество Adm:

— $Adm \subseteq (R \cup AR) \times (R \cup AR)$ — множество ролей, доступных для делегирования членства, при этом без административного права $createrole_r$ может быть передано только членство в ролях, полученное с опцией *WITH_ ADMIN_ OPTION*.

Управление ролями разрешено только роли, имеющей административное право доступа $createrole_r$ или получившей членство с опцией *WITH_ ADMIN_ OPTION*. Управление ролями, имеющими право доступа $superuser_r$, разрешено только ролям, в свою очередь имеющим это право доступа. При отборе членства возможен отбор опции *WITH_ ADMIN_ OPTION*.

Как было сказано ранее, для предотвращения неконтролируемого распространения прав доступа предлагается разделить роли SQL на административные и неадминистративные. При этом административные роли могут быть использованы только доверенными субъект-сессиями. Это отличие от требований SQL влечёт необходимость доработки соответствующих механизмов управления ролями в СУБД.

Особенностью реализации наследования прав доступа в иерархии ролей в СУБД PostgreSQL является наличие признака, разрешающего или запрещающего подобное наследование. Права доступа наследуются только в случае наличия у роли свойства *INHERIT*. Для описания подобного поведения введена функция $role\_inherit$:

— $role\_inherit : R \cup AR \to \{\texttt{true}, \texttt{false}\}$ — функция наследования прав доступа, задающая для каждой роли режим их наследования; если значение функции равно `true`, то роль, стоящая в иерархии выше текущей, наследует её права доступа, иначе нет. При этом $role\_inherit(public) = \texttt{true}$.

Для упрощения выражений в правилах преобразования предлагается ввести следующие обозначения:

— $H_R^{\sim} : R \cup AR \to 2^R \cup 2^{AR}$ — рекурсивная функция доступных ролей с учётом наследования прав доступа, удовлетворяющая для каждой роли $r \in R \cup AR$ условию: если значение функции $role\_inherit(r) = \texttt{true}$, то $H_R^{\sim}(r) = \{r\} \cup \bigcup\limits_{r' \in H_R(r)} H_R^{\sim}(r')$, иначе $H_R^{\sim}(r) = \{r\}$;

— $PA^{\sim} : R \cup AR \to 2^P$ — рекурсивная функция прав доступа к сущностям ролей и административных ролей с учётом иерархии, задающая множество всех доступных роли прав доступа, удовлетворяющая для $r \in R \cup AR$ условию: если значение функции $role\_inherit(r) = \texttt{true}$, то $PA^{\sim}(r) = PA(r) \cup \bigcup\limits_{r' \in H_R(r)} PA^{\sim}(r')$, иначе $PA^{\sim}(r) = PA(r)$.

Функция $PA^{\sim}$ позволяет получить всё множество доступных заданной роли прав доступа.

По аналогии с МРОСЛ ДП-моделью, для краткости и удобства описания элементов модели с учётом мандатного управления доступом вводится функция доступа субъект-сессии к сущности в контейнере с учётом прав доступа к сущностям-контейнерам, её содержащим. Но, в отличие от ОС, в СУБД у контейнеров отсутствует право доступа *execute*. Таким образом, задана следующая функция:

— $lookup : S \times E \setminus O_d \to \{\texttt{true}, \texttt{false}\}$ — функция доступа субъект-сессии к сущности в контейнере с учётом прав доступа к сущностям-контейнерам, её содержащим, такая, что по определению для субъект-сессии $s \in S$ и сущности $e \in E \setminus O_d$ справедливо равенство $lookup(s, e) = \texttt{true}$ тогда и только тогда, когда $e \in C_{cl}$ (доступ субъект-сессии к кластеру по определению есть) или если $e \in E \setminus O_d$ и существует последовательность сущностей $e_0, \ldots, e_n \in E$, где $n \geqslant 1$, $e_0 \in C_{cl}$, $e = e_n$, удовлетворяющих следующим условиям:
  — $e_i \in H_E(e_{i-1})$, где $1 \leqslant i \leqslant n$;
  — существует $r_i \in R \cup AR$, такая, что $(s, r_i, read_a) \in AA$ и
  если $e_i \in C_{db}$, то $(e_i, connect_r) \in PA\ (r_i)$,
  если $e_i \in C_{nsp}$, то $(e_i, usage_r) \in PA\ (r_i)$,
  для других типов контейнеров дополнительных условий не требуется;
  — $f_e(e_i) \leqslant f_s(s)$, или $CCR(e_i) = \texttt{false}$, или $downgrade\_admin\_sop \in session\_option(x)$.

Объекты-данные O_d исключаются, так как доступ к ним регламентируется отдельными правами доступа и осуществляется только через другие объекты.

Определим $G = (UA, AUA, PA, user, role, FC, CCR, A, AA, F, H_R, H_E, L_U, L_S)$ — состояние системы; используем обозначения: $\sum(G^*, OP)$ — система, при этом:

— G^* — множество всех возможных состояний;
— OP — множество правил преобразования состояний;
— $G \vdash_{op} G'$ — переход системы $\sum(G^*, OP)$ из состояния G в состояние G' с использованием правила преобразования состояний $op \in OP$.

В начальном состоянии $G_0 = (UA_0, AUA_0, PA_0, user_0, role_0, FC_0, CCR_0, A_0, AA_0, F_0, H_{R_0}, H_{E_0}, L_{U_0}, L_{S_0})$ по определению справедливо $A_0 = \varnothing$, $AA_0 = \varnothing$, $F_0 = \varnothing$.

3. Правила преобразования состояний

Далее приведено описание основных правил преобразования состояний системы, заданных в рамках PostgreSQL ДП-модели, в которых по существу используются её новые элементы. Кроме того, на реализации именно этих правил сконцентрированы усилия автора при внедрении модели в рассматриваемую СУБД ОС СН.

Для упрощения описания правил преобразования будем записывать только те составляющие состояния G', которые меняются относительно состояния G.

Наличие права $superuser_r$ разрешает любые действия и для упрощения без необходимости не указывается.

Де-юре правила преобразования состояний PostgreSQL ДП-модели

Правило	Исходное состояние G	Результирующее состояние G'
1	2	3
$access\_read(x, y)$	$x \in S$, $y \in (E \setminus O_d) \cup R \cup AR$, [$y \in E \setminus O_d$, (либо $lookup(x, y) =$ `true` и $f_e(y) \leqslant f_s(x)$, либо $downgrade\_admin\_sop \in session\_option(x)$)], [$y \in R \cup AR$, (либо $f_r(y) \leqslant f_s(x)$, либо $downgrade\_admin\_sop \in session\_option(x)$)]	если $y \in E \setminus O_d$, то $A' = A \cup \{(x, y, read_a)\}, AA' = AA$, если $y \in R \cup AR$, то $AA' = AA \cup \{(x, y, read_a)\}, A' = A$
$access\_write(x, y)$	$x \in S$, $y \in (E \setminus O_d) \cup R \cup AR$, [$y \in E \setminus O_d$, (если $y \in C_{cl}$, то $(y, superuser_r) \in PA^{\sim}(role(x))$, если $y \in C_{db} \cup C_{nsp}$, то $(y, create_r) \in PA^{\sim}(role(x))$), (либо $lookup(x, y) =$ `true` и ($f_e(y) = f_s(x)$ или ($f_s(x) < f_e(y)$ и $CCR(y) =$ `false`)), либо $downgrade\_admin\_sop \in session\_option(x)$)], [$y \in R \cup AR$, $(y, createrole_r) \in PA^{\sim}(role(x))$, (либо $f_r(y) = f_s(x)$, либо $downgrade\_admin\_sop \in session\_option(x)$)]	если $y \in E \setminus O_d$, то $A' = A \cup \{(x, y, write_a)\}, AA' = AA$, если $y \in R \cup AR$, то $AA' = AA \cup \{(x, y, write_a)\}, A' = A$
$access\_append(x, y)$	$x \in S$, $y \in (E \setminus O_d)$, (если $y \in C_{cl}$, то $(y, superuser_r) \in PA^{\sim}(role(x))$, если $y \in C_{db} \cup C_{nsp}$, то $(y, create_r) \in PA^{\sim}(role(x))$], (либо $f_s(x) < f_e(y)$, либо $downgrade\_admin\_sop \in session\_option(x)$))	$A' = A \cup \{(x, y, read_a)\}, AA' = AA$
$delete\_access(x, y, \alpha_a)$	$x \in S, y \in E \cup R \cup AR$, $(x, y, \alpha_a) \in A \cup AA$	если $y \in E$, то $A' = A \setminus (x, y, \alpha_a), AA' = AA$, если $y \in R \cup AR$, то $AA' = AA \setminus (x, y, \alpha_a), A' = A$
$access\_select(x, y, \{col_j : 1 \leqslant j \leqslant k\}, D_{out})$	$x \in S$, $y \in C_{rel}$, $(x, y, read_a) \in A$, $\{col_j : 1 \leqslant j \leqslant k\} \in O_{col} \cap H_E(y)$, $D_{out} \subseteq O_d \cap H_E(y)$, (либо $(y, select_r) \in PA^{\sim}(role(x))$, либо $(col_j, select_r) \in PA^{\sim}(role(x))$, $1 \leqslant j \leqslant k$), (либо $\forall t \in D_{out}(f_e(t) \leqslant f_s(x))$, либо $downgrade\_admin\_sop \in session\_option(x)$)	$A' = A \cup \{(x, t, read_a) : t \in D_{out}\}$, $\forall er \in rules(y, select_r)$ выполняется $execute\_rule(er)$

Продолжение таблицы

1	2	3
$access\_insert(x, y, D_{in})$	$x \in S, y \in C_{rel}, (x, y, append_a) \in A$, $D_{in} \not\subseteq O$, (либо $(y, insert_r) \in PA^{\sim}(role(x))$, либо $(c, insert_r) \in PA^{\sim}(role(x))$, $c \in \{c : c \in H_E(y), type(c) = col\}$)	$A' = A \cup \{(x, t, append_a) : t \in D_{in}\}$, $\forall t \in D_{in}\ f'_e(t) = f_s(x)$, $O'_d = O_d \cup D_{in}, H'_E(y) = H_E(y) \cup D_{in}$, $\forall er \in rules(y, insert_r)$ выполняется $execute\_rule(er)$, $\forall et \in triggers(y, insert_r)$ выполняется $execute\_procedure(trigger\_proc(et))$
$access\_update(x, y, \{col_j : 1 \leqslant j \leqslant k\}, D_{mod})$	$x \in S$, $y \in C_{rel}$, $(x, y, write_a) \in A$, $\{col_j : 1 \leqslant j \leqslant k\} \in O_{col} \cap H_E(y)$, $D_{mod} \subseteq O_d \cap H_E(y)$, (либо $(y, update_r) \in PA^{\sim}(role(x))$, либо $(col_j, update_r) \in PA^{\sim}(role(x))$, $1 \leqslant j \leqslant k$), (либо $\forall t \in D_{mod}(f_e(t) = f_s(x))$, либо $downgrade\_admin\_sop \in session\_option(x)$)	$A' = A \cup \{(x, t, write_a) : t \in D_{mod}\}$, $\forall er \in rules(y, update_r)$ выполняется $execute\_rule(er)$, $\forall et \in triggers(y, update_r)$ выполняется $execute\_procedure(trigger\_proc(et))$
$access\_delete(x, y, D_{del})$	$x \in S$, $y \in C_{rel}$, $(x, y, write_a) \in A$, $D_{del} \subseteq O_d \cap H_E(y)$, (либо $(y, delete_r) \in PA^{\sim}(role(x))$, либо $(c, delete_r) \in PA^{\sim}(role(x))$, $c \in \{c : c \in H_E(y), type(c) = col\}$), (либо $\forall t \in D_{del}(f_e(t) = f_s(x))$, либо $downgrade\_admin\_sop \in session\_option(x)$)	$O'_d = O_d \setminus D_{del}$, $H'_E(y) = H_E(y) \setminus D_{del}$, $\forall er \in rules(y, delete_r)$ выполняется $execute\_rule(er)$, $\forall et \in triggers(y, delete_r)$ выполняется $execute\_procedure(trigger\_proc(et))$, $A' = A \setminus \{(s, t, \alpha_a) : s \in S, t \in D_{del}, \alpha_a \in R_a\}$
$create\_rel(x, y, z, Q_{col}, Q_{cnstr})$	$x \in S, y \notin E, z \in C_{nsp}, type(y) = rel$, $\forall e \in Q_\alpha\ type(e) = \alpha$ и $e \notin E$, $(x, z, append_a) \in A$, $\forall e' \in Q_{col}\ (x, coltype(e'), read_a) \in A$	$E' = E \cup \{y\} \cup Q_{col} \cup Q_{cnstr}$, $(C' = C \cup \{y\}, O' = O \cup Q_{col} \cup Q_{cnstr})$, $H'_E(z) = H_E(z) \cup \{y\}$, $H'_E(y) = Q_{col} \cup Q_{cnstr}$, $f'_e(y) = f_s(x), CCR'(y) = \texttt{true}$, $PA'(role(x)) = PA(role(x)) \cup ALL_r(y)$, $Gr'(role(x)) = Gr(role(x)) \cup ALL_{r(y)}$
$create\_procedure(x, y, z, as\_option, \{op_j : 1 \leqslant j \leqslant k\})$	$x \in S, y \notin E, z \in C_{nsp}$, $as\_option \in \{as\_caller, as\_owner\}$, $\{op_j : 1 \leqslant j \leqslant k\} \subseteq OP$, $(x, z, append_a) \in A$	$O' = O \cup \{y\}$, $(O'_{proc} = O_{proc} \cup \{y\}$, $C' = C)$, $H'_E(z) = H_E(z) \cup \{y\}$, $operations'(y) = \{op_j : 1 \leqslant j \leqslant k\}$, $f'_e(y) = f_s(x)$, $execute\_as'(y) = as\_option$, $PA'(role(x)) = PA(role(x)) \cup ALL_r(y)$, $Gr'(role(x)) = Gr(role(x)) \cup ALL_r(y)$, $PA'(public) = PA(public) \cup \{(y, execute_r)\}$
$execute\_procedure(x, y)$	$x \in S, y \in O_{proc}, (x, y, read_a) \in A$, $(y, execute_r) \in PA^{\sim}(role(x))$	если $execute\_as(y) = as\_owner$, то: 1) $\exists r \in R \cup AR$ $(y, own_r) \in PA(r)\ role'(x) = r$; 2) выполняется $G \vdash_{op_1} \ldots \vdash_{op_n} G_n = G'$, $op_i \in operations(x), i = 1, \ldots, n$; 3) $role'(x) = role(x)$, иначе выполняется $G \vdash_{op_1} \ldots \vdash_{op_n} G_n = G'$, $op_i \in operations(x), 1 \leqslant i \leqslant n$

Окончание таблицы

1	2	3
$create\_trigger(x, y, z, p, \{r_j : 1 \leqslant j \leqslant k\})$	$x \in S$, $y \notin E$, $z \in C_{rel}$, $p \in O_{proc}$, $\{r_j : 1 \leqslant j \leqslant k\} \subseteq \{insert_r, update_r, delete_r, truncate_r\}$, $(x, z, write_a) \in A, (x, p, read_a) \in A$, $(z, trigger_r) \in PA^{\sim}(role(x))$	$O' = O \cup \{y\}, (O'_{trg} = O_{trg} \cup \{y\}$, $C' = C), H'_E(z) = H_E(z) \cup \{y\}$, $f'_e(y) = f_s(x), trigger\_proc'(y) = p$, $triggers'(z, r_j) = triggers(z, r_j) \cup \{y\}$, $1 \leqslant j \leqslant k$
$create\_rule(x, y, z, op, r)$	$x \in S$, $y \notin E$, $z \in C_{rel}$, $op \in \{access\_select, access\_insert, access\_update, access\_delete\}$, $r \in \{select_r, insert_r, update_r, delete_r\}$, $(x, z, write_a) \in A$, $(z, own_r) \in PA^{\sim}(role(x))$	$O' = O \cup \{y\}, (O'_{rule} = O_{rule} \cup \{y\}$, $C' = C), H'_E(z) = H_E(z) \cup \{y\}$, $f'_e(y) = f_s(x), rule\_op'(y) = op$, $rules'(z, r) = rules(z, r) \cup \{y\}$
$execute\_rule(x, y, z, r)$	$x \in S$, $y \in O_{rule}$, $y \in H_E(z)$, $r \in \{select_r, insert_r, update_r, delete_r\}$, $(x, z, \alpha) \in A$, ($\alpha = read_a$ для $r = select_r$, $\alpha = append_a$ для $r = insert_r$, $\alpha \in \{read_a, write_a\}$ для $r \in \{update_r, delete_r\}$)	выполняется $G \vdash_{op(y)} G_{op(y)} = G'$
$grant\_rights(x, r, \{(y, \alpha_{rj}) : 1 \leqslant j \leqslant k\}, grant\_option)$	$x \in S$, $y \in E \setminus (O_d \cup C_{cl})$, $r \in R \cup AR$, $\{(y, \alpha_{rj}) : 1 \leqslant j \leqslant k\} \subseteq ALL_r(y)$, $\{(y, \alpha_{rj}) : 1 \leqslant j \leqslant k\} \subseteq Gr(role(x))$, $grant\_option \in \{\texttt{true}, \texttt{false}\}$, $(x, r, write_a) \in AA$	$PA'(r) = PA(r) \cup \{(y, \alpha_{rj}) : 1 \leqslant j \leqslant k\}$, (если $r \neq public$ и $grant\_option = \texttt{true}$, то $Gr'(r) = Gr(r) \cup \{(y, \alpha_{rj}) : 1 \leqslant j \leqslant k\}$)
$revoke\_rights(x, r, \{(y, \alpha_{rj}) : 1 \leqslant j \leqslant k\}, grant\_option)$	$x \in S$, $y \in E \setminus (O_d \cup C_{cl})$, $r \in R \cup AR$, $\{(y, \alpha_{rj}) : 1 \leqslant j \leqslant k\} \subseteq ALL_r(y)$, $\{(y, \alpha_{rj}) : 1 \leqslant j \leqslant k\} \subseteq Gr(role(x))$, $grant\_option \in \{\texttt{true}, \texttt{false}\}$, $(x, r, write_a) \in AA$	[если $r \neq public$, то $Gr'(r) = Gr(r) \setminus \{(y, \alpha_{rj}) : 1 \leqslant j \leqslant k\})$], [если $r = public$ или $grant\_option = \texttt{false}$, то $PA'(r) = PA(r) \setminus \{(y, \alpha_{rj}) : 1 \leqslant j \leqslant k\}$]

Де-юре правила $access\_read(x, y)$, $access\_write(x, y)$ и $access\_append(x, y)$ аналогичны одноименным правилам МРОСЛ ДП-модели и позволяют субъект-сессии x получить соответствующий доступ к сущности y с определёнными условиями на уровень конфиденциальности субъект-сессии. Отличие заключается в замене функции $execute\_container(x, y)$ на функцию $lookup(x, y)$. Кроме того, при доступе к кластеру необходимо наличие у роли субъект-сессии доступного по иерархии права $superuser_r$, а при доступе к базам данных и схемам — наличие права $create_r$ к ним. Для нарушения требований к уровню конфиденциальности субъект-сессии требуется наличие у субъект-сессии параметра $downgrade\_admin\_sop$. Де-юре правило $delete\_access(x, y, \alpha_a)$ позволяет субъект-сессии x, обладающей доступом α_a к сущности, роли или административной роли y, удалить этот доступ.

Де-юре правила $access\_select(x, y, \{col_j : 1 \leqslant j \leqslant k\}, D_{out})$, $access\_insert(x, y, D_{in})$, $access\_update(x, y, \{col_j : 1 \leqslant j \leqslant k\}, D_{mod})$ и $access\_delete(x, y, D_{del})$ отражают специфику работы с данными в таблицах БД и позволяют субъект-сессии x получить доступ на выборку, вставку, изменение и удаление данных D сущности-таблицы y. Для выполнения операции необходимо наличие у роли субъект-сессии x доступных по иерархии необходимых доступов к таблице и соответствующего операции права доступа. Операции затрагивают только доступные по мандатным атрибутам записи таблицы (для нарушения указанного поведения требуется наличие у субъект-сессии параметра $downgrade\_admin\_sop$). При выполнении операции вызываются на исполнение соответствующие реализованному праву доступа триггеры и правила, заданные для указанной таблицы функциями $triggers$ и $rules$ соответственно.

Правила создания контейнеров рассмотрим на примере де-юре правила создания таблицы $create\_rel(x, y, z, Q_{col}, Q_{cntsr})$, которое позволяет субъект-сессии СУБД x создать новую контейнер-таблицу y в схеме z с указанным множеством столбцов Q_{col} и ограничений Q_{cnstr}. Для этого требуется наличие у субъект-сессии x доступа на добавление к схеме z и на чтение к сущностям-типам, указанным для столбцов из Q_{col}. При этом новая сущность-контейнер приобретает уровень конфиденциальности, равный уровню доступа субъект-сессии x, и мандатный атрибут CCR, равный `true`. Роль субъект-сессии получает все права доступа к новой сущности с опцией их делегирования (множество Gr).

Де-юре правило $create\_procedure(x, y, z, as\_option, \{op_j : 1 \leqslant j \leqslant k\})$ позволяет субъект-сессии x создать процедуру y в контейнере $z \in C_{nsp}$ при наличии к нему доступа на добавление. Для создаваемой процедуры указывается режим выполнения кода $as\_option \in \{as\_caller, as\_owner\}$ и упорядоченный конечный набор операций $\{op_j : 1 \leqslant j \leqslant k\}$, входящих в множество OP правил преобразования состояния модели. При этом новая сущность приобретает уровень конфиденциальности, равный уровню доступа субъект-сессии x, указанный режим исполнения кода, а в качестве кода — состав указанных операций. Роль субъект-сессии получает все права доступа к новой сущности с опцией их делегирования (множество Gr). Для процедур по умолчанию предоставляется право на исполнение $execute_r$ группе $PUBLIC$.

Де-юре правило $execute\_procedure(x, y)$ позволяет субъект-сессии x выполнить процедуру y при наличии к ней доступа на чтение и доступного по иерархии права исполнения $execute_r$ для указанной процедуры у роли субъект-сессии. В ходе исполнения процедуры согласно упорядоченному набору операций, составляющих тело процедуры $operations$, последовательно выполняются преобразования состояния модели. При этом если для процедуры указан режим исполнения кода as_owner, перед выполнением процедуры текущая роль субъект-сессии меняется на роль, обладающую правом владения к процедуре; после исполнения исходная роль субъект-сессии восстанавливается.

Де-юре правила $create\_trigger(x, y, z, p, \{r_j : 1 \leqslant j \leqslant k\})$ и $create\_rule(x, y, z, op, r)$ позволяют субъект-сессии x создать для таблицы $z \in C_{rel}$ новый триггер или правило, выполняющиеся при реализации прав доступа $\{r_j : 1 \leqslant j \leqslant k\} \subseteq \{insert_r, update_r, delete_r, truncate_r\}$ или $r \in \{select_r, insert_r, update_r, delete_r\}$ соответственно. При создании триггера указывается активируемая процедура $p \in O_{proc}$, причём необходимо наличие у субъект-сессии z доступов на запись к таблице z и чтение к процедуре p и доступного по иерархии права создания триггера $trigger_r$ для указанной таблицы у роли субъект-сессии. При создании правила указывается операция op из перечисленных $\{access\_select, access\_insert, access\_update, access\_delete\} \in OP$ правил преобразования состояния модели, причём необходимо наличие у субъект-сессии x доступного по иерархии права владения к таблице z. Новая сущность приобретает уровень конфиденциальности, равный уровню доступа субъект-сессии x, помещается в иерархию таблицы z, обновляются функции $triggers$ и $rules$ соответственно; устанавливается связь между триггером и активируемой процедурой или правилом таблицы и правилом преобразования состояния модели.

Де-юре правило $execute\_rule(x, y, z, r)$ позволяет субъект-сессии x выполнить правило y таблицы z при реализации права доступа r. Данное правило вызывается автоматически при реализации заданного права доступа к таблице. Для этого необходимо наличие соответствующего доступа к таблице. При выполнении этого правила реально исполняется указанное при создании правило op преобразования состояния модели.

Де-юре правила $grant\_rights(x, r, \{(y, \alpha_{rj}) : 1 \leqslant j \leqslant k\}, grant\_option)$ и $revoke\_rights(x, r, \{(y, \alpha_{rj}) : 1 \leqslant j \leqslant k\}, grant\_option)$ позволяют субъект-сессии x добавить или удалить соответственно права доступа к сущности y из множества прав доступа роли или административной роли r. Для применения правил необходимо наличие у субъект-сессии x доступа на запись к роли r и перечисленных прав доступа в списке делегирования Gr роли субъект-сессии. Параметр $grant\_option$ соответствует опции *SQL WITH_ GRANT_ OPTION*, предоставляющей роли r возможность делегировать полученные права. Данная опция не может быть применена к встроенной группе *PUBLIC*. При удалении, если параметр $grant\_option$ установлен в `true`, отбирается только возможность делегирования.

Де-факто правила, используемые для отражения факта получения субъект-сессией де-факто владения субъект-сессиями или факта реализации информационного потока по памяти или по времени, схожи с аналогичными правилами МРОСЛ ДП-модели.

Заключение

Основной целью адаптации МРОСЛ ДП-модели для использования в реляционной СУБД PostgreSQL и разработки новой PostgreSQL ДП-модели является обеспечение единого подхода к управлению доступом в СУБД и ОС СН Astra Linux Special Edition.

Проводимое исследование еще не закончено, однако уже в настоящее время в рамках PostgreSQL ДП-модели удалось описать специфичные для исследуемой СУБД элементы, отличные от заданных в известных автору моделях управления доступом в ОС или СУБД. При этом наибольшее внимание уделено учёту специфики реляционных баз данных и сохранению совместимости с традиционными требованиями стандарта SQL.

Создание модели позволит не только сформулировать и обосновать условия нарушения безопасности в СУБД, но и проверить их корректность в реальной системе. В дальнейшем интеграция СУБД с реализацией PostgreSQL ДП-модели в разрабатываемую версию ОС СН позволит повысить защищённость информации в автоматизированных системах, создаваемых на их основе.

ЛИТЕРАТУРА

1. *Гостехкомиссия России.* Руководящий документ. Средства вычислительной техники. Защита от несанкционированного доступа к информации. Показатели защищённости от несанкционированного доступа к информации. М.: Военное издательство, 1992.
2. *Гостехкомиссия России.* Руководящий документ. Автоматизированные системы. Защита от несанкционированного доступа к информации. Классификация автоматизированных систем и требования по защите информации. М.: Военное издательство, 1992.
3. *Девянин П. Н.* О разработке мандатной сущностно-ролевой ДП-модели управления доступом и информационными потоками в операционных системах семейства Linux // Методы и технические средства обеспечения безопасности информации: материалы 21-й науч.-технич. конф. 24–29 июня 2012 г. СПб.: Изд-во Политехн. ун-та, 2012. С. 91–94.
4. *Девянин П. Н.* Об опыте внедрения мандатной сущностно-ролевой ДП-модели управления доступом и информационными потоками в защищенную ОС Astra Linux Special Edition // Методы и технические средства обеспечения безопасности информации: материалы 22-й науч.-технич. конф. 08–11 августа 2013 г. СПб.: Изд-во Политехн. ун-та, 2013. С. 78–80.
5. Операционные системы Astra Linux [Электронный ресурс]. `http://www.astra-linux.ru/`.
6. *Sandhu R.* Rationale for the RBAC96 family of access control models // Proc. 1st ACM Workshop on Role-Based Access Control. ACM, 1997.

7. *Девянин П. Н.* Модели безопасности компьютерных систем. Управление доступом и информационными потоками: учеб. пособие для вузов. М.: Горячая линия-Телеком, 2011. 320 с.
8. *Колегов Д. Н.* О построении иерархического ролевого управления доступом // Прикладная дискретная математика. 2012. № 5. С. 69–71.
9. *Смольянинов В. Ю.* О достаточных условиях похищения прав доступа в СУБД ДП-модели // Прикладная дискретная математика. 2012. № 5. С. 75–76.
10. *Смирнов С. Н.* Безопасность систем баз данных. М.: Гелиос АРВ, 2007. 352 с.
11. PostgreSQL 9.2.1 Documentation. The PostgreSQL Global Development Group, 2012. 2605 p.
12. Information technology — Database languages — SQL — Part 1: Framework (SQL/Framework). ISO/IEC 9075:1999, 1999.
13. Information technology — Database languages — SQL — Part 2: Foundation (SQL/Foundation). ISO/IEC 9075:1999, 1999.
14. *Sumathi S. and Esakkirajan S.* Fundamentals of Relational Database Management Systems. Springer, 2007. 776 p.
15. *Lane T.* A Tour of PostgreSQL Internals. Great Bridge, LLC, 2000. 25 p. [Электронный ресурс.] `http://www.postgresql.org/files/developer/tour.pdf`.
16. *Шумилин А. В.* Перспективный подход к обеспечению защиты информации от несанкционированного доступа в СУБД // Новые технологии. Программная инженерия. 2012. № 1. С. 35–40.
17. *Шумилин А. В.* Подход к оцениванию влияния средств разграничения доступа к данным на производительность реляционных СУБД // Новые технологии. Программная инженерия. 2013. № 4. С. 29–33.
18. *Шумилин А. В.* Применение мандатной сущностно-ролевой ДП-модели управления доступом и информационными потоками в СУБД операционной системы специального назначения Astra Linux Special Edition // Методы и технические средства обеспечения безопасности информации: материалы 22-й науч.-технич. конф. 08–11 августа 2013 г. СПб.: Изд-во Политехн. ун-та, 2013. С. 87–89.