УДК 681.3.06

АВТОМАТИЗИРОВАННОЕ СОЗДАНИЕ ПРАВИЛ УПРАВЛЕНИЯ ДОСТУПОМ К ДАННЫМ СРЕДСТВАМИ СУБД

А.А. Блажко, Ибаа Сауд

Одесский национальный политехнический университет, 65044, Одесса, проспект Шевченко, 1,
blazhko@ieee.org

Прикладные программы доступа к базам данных в корпоративной информационной системе с целью обеспечения гибкости политики безопасности при доступе к данным требуют управления доступом через программирование механизма доступа на уровне строк и столбцов таблиц БД (Row Level Security). Рост числа пользователей и таблиц в БД увеличивает сложность этого процесса управления. Предлагается метод автоматизированного создания правил управления доступом к данным программными средствами активных СУБД для сокращения числа операций при создании пользовательских пространств. Предложен алгоритм для автоматического создания SQL-запросов Row Level Security механизма, который подходит для большинства СУБД, использующих избирательное управление доступом. Метод использует структурно-должностную иерархию пользователей, словари базы данных и программные шаблоны операций управления доступом в различных СУБД.

Database applications in enterprise information system for flexibility of security policy large require the Row Level Security mechanism. Large number of users and tables in database increases the process complexity of administration. In this paper, we propose automated design method of hierarchical access control in database to reduce the number of operations for user data spaces creation. An algorithm for automatic creation of SQL-queries in the Row Level Security, which is suitable for most databases using the Discretionary Access Control, is proposed. Method uses structural-post hierarchy users, database dictionary and templates of access control commands for different DBMS.

Введение

Большинство корпоративных информационных систем (ИС) хранит данные о деятельности организации в базе данных (БД), управляемой СУБД с использованием языка *SQL*. При этом пользователи ИС должны иметь доступ только к данным в соответствии с их официальными обязанностями в организации. Это реализуется через описание политики безопасности при управлении доступом (УД), в которой объекты доступа – данные в БД, субъекты доступа – пользователи или программы (процессы) ИС [1]. В основном используются три модели УД: полномочное управление доступом (*MAC – Mandatory Access Level*), избирательное управление доступом (*DAC – Discretionary Access Level*) и ролевое управление доступом (*RBAC – Role-based Access Level*) [2].

В соответствии с ограничениями доступа к данным для пользователей объект доступа должен рассматриваться на разных уровнях грануляции (детализации): множество таблиц, подмножество записей таблицы, подмножество атрибутов таблицы, ячейка данных [3]. Прозрачность управления доступа для пользователя осуществляется с использованием гранулированного доступа и реализуется через программирование на уровне строк и столбцов таблиц БД в виде *Row Level Security* [4] или *Fine-grained* УД [5–7]. *SQL*-язык предоставляет ограниченные формы *DAC* в БД, так как он представляет объект доступа только в виде базовой таблицы или виртуального представления [5]. Детализация уровня доступа на уровне строк таблицы создается только через дополнительные нестандартные механизмы, предоставляемые промышленными СУБД (*Oracle, IBM DB2, MS SQL, Sybase* и др.) или *Opensource* СУБД (например, *PostgreSQL*).

Например, СУБД *Oracle* предлагает специальный *RLS*-механизм для *DAC, MAC* с использованием виртуальных частных баз данных (*VPD – Virtual Private Databases*), *IBM DB2* – для *MAC, Sybase* – для *DAC*. Но этот механизм требует трудоемкого процесса программирования для каждого пользователя-субъекта и таблицы-объекта. В работе [5] показано как динамический мета-уровень УД может быть использован через модификацию *SQL*-запросов. Но в работе не используются описания операций УД конкретной СУБД, а также отмечено, что для адаптации пользовательских представлений необходимы дополнительные сложные настройки. В работе [6] представлены авторизованные представления, которые обеспечивают прозрачное выполнение запросов за счет введения новых определений корректности и условий корректности запросов. При этом отмечено, что процесс создания пользовательских представлений не масштабируется при большом количестве пользователей. В работе [7] предлагается меточный подход для маскирования неавторизованной информации, а также алгоритм оценки несанкционированности запросов. Но он использует промежуточное программное обеспечение между пользователем и СУБД для выполнения переписанного запроса для СУБД *Oracle*. Авторы указывают на сложность задач администрирования при использовании гетерогенных распределенных БД [8]. В этом случае для каждой БД необходимо использовать операции управления доступом конкретной СУБД.

С учетом вышеизложенного для уменьшения трудоемкости процесс администрирования необходимы:

1) унификация всех предложенных методов УД при использовании операций управления доступом для каждой СУБД;

2) автоматизация процесса программирования кода при создании *VDP/RLS*-механизмов.

Унификация может уменьшить сложность процесса управления в ИС при гетерогенности СУБД и возросшем количестве таблиц и пользователей. Унификация процесса УД чаще всего основывается на использовании шаблонов безопасности. В работе [9] шаблоны безопасности представляются в структурированном виде для *DAC* и *RBAC*: назначение, контекст, проблемы, решения. Определяется концептуальный *UML* для создания моделей УД, но не используются детали операций конкретных СУБД.

Автоматизация возможна только при учете характеристик ИС, но ее эффективность увеличивается при уменьшении количества характеристик ИС. Примером может быть структура организации [10, 11]. Большинство ИС используют иерархию пользователей, которая соответствует структурно-должностной иерархии сотрудников в организации. Автоматизация процесса создания программного кода может эффективно использовать такие метаданные.

Автоматизированное создание правил управления доступом к данным средствами СУБД

В данной работе предлагается метод автоматизированного создания правил управления доступом к данным средствами СУБД. Метод включает четыре этапа.

Этап 1. Автоматическое формирование субъектов на основе содержимого БД по сотрудникам организации.

Этап 2. Автоматическое определение уровня доступа субъектов на основе структурно-должностной иерархии.

Этап 3. Автоматическое формирование представления субъектов об объектах доступа на основе словаря данных и описания субъектов.

Этап 4. Автоматическое создание правил управления доступом на основе механизма активных БД, который основан на правилах СУБД (триггерах).

На рис. 1 представлены этапы автоматизированного создания правил управления доступом к данным средствами СУБД.

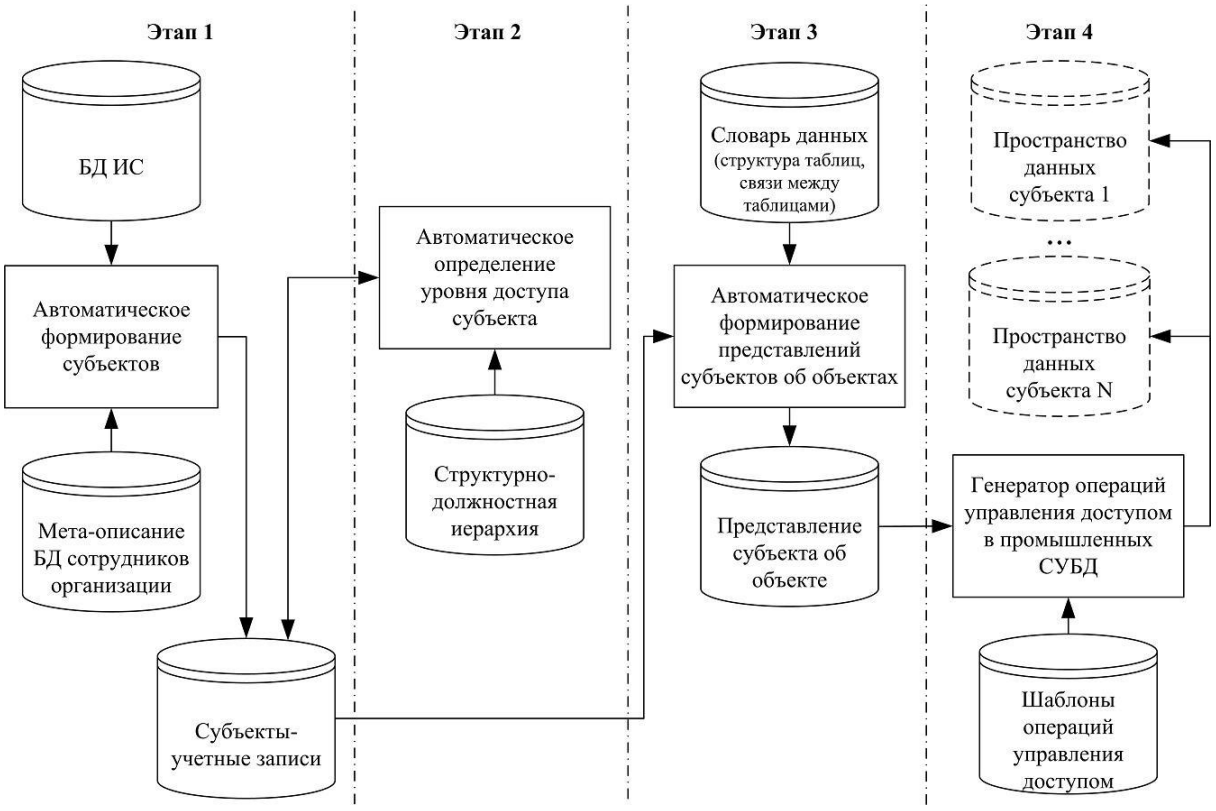

Рис.1. Этапы автоматизированного создания правил управления доступом к данным средствами СУБД

Первый этап снижает трудоемкость процесса создания базы данных пользователей как сотрудников организации за счет формализации информации о сотрудниках в структуре базы данных. Такая формализация представлена в виде кортежа *user_meta_data = <attr_name, attr_def>,*

где *attr_name* – название атрибута таблицы учетных записей пользователей;

attr_def – множество предикатов как путь доступа к атрибутам таблиц БД ИС, которые содержат данные по атрибуту таблицы учетных записей пользователей.

Предложенная формализация позволяет автоматически создавать учетные записи пользователей, которые включают: атрибут-ссылку на структуру БД, имя пользователя, описание пользователя как сотрудника организации. Предусмотрено создание функции преобразования данных о пользователе в его уникальное имя на латинском языке.

На втором этапе для каждой учетной записи пользователя-субъекта определяется его уровень доступа к данным на основе принятой в организации структурно-должностной иерархии. Для автоматического определения уровня доступа такая иерархия должна устанавливать связь со структурой БД ИС, поэтому предложено ее представлять в виде кортежа < *l, pd, dd, pl*>,
где *l* – числовое значение уровня иерархии;

pd – множество предикатов описания должности сотрудника в БД ИС как часть *where*-фразы в *SQL*-запросах;

dd – множество предикатов описания подразделения сотрудника в БД ИС как часть *where*-фразы в *SQL*-запросах;

pl – числовое значение родительского уровня иерархии.

Третий этап автоматически формирует представления субъектов об объектах доступа на основе словаря данных и описания учетных записей пользователей-субъектов.

Предлагается алгоритм создания представления субъекта с именем *UName* об объекте *Tname*, включающий такую последовательность шагов.

Шаг 1. Получение *col_name* как первичного ключа таблицы *Tname*.

Шаг 2. Получение *UserMetaData.db_link, UserMetaData.name*

Шаг 3. *own_user_view* := '*select* ' + *col_name* + ' *from* ' + *Tname* + ' *where* '.

Шаг 4. Если *col_name* <> *UserMetaData.db_link*, то *own_user_view* := *own_user_view* + + *short_path*(*col_name, UserMetaData.db_link*).

Шаг 5. *own_user_view* := *own_user_view* + *UserMetaData.name* + '=' + *Uname.*

Шаг 6. Создание массива *SQL*-запросов *child_user_view[]*, которые получают данные из таблицы *Tname*, относящиеся к пользователям, являющимися подчиненными пользователя *Uname*, используя содержимое таблицы *HierarhieLevel*. Для получения данных в запросы включаются подзапросы по квалификатору *exists*.

Шаг 7. Формирование запроса представления пользователя: *user_view* = ‘(’ + *own_user_view* + ‘)*union*(’ + + *child_user_view*[1] + ‘)*union*(’ + … *child_user_view*[*N*] + ‘)‘

В процессе создания *SQL*-запросов необходимо устанавливать связь между отдельными атрибутами таблиц БД ИС. Для этого разработан алгоритм установления такой связи на основе одного из алгоритмов поиска кратчайшего пути на графе. Узлами графа являются атрибуты таблиц БД ИС. Дуги графа имеют веса, которые принимают два значения: 1 – атрибуты узлов принадлежат одной таблице, 2 – атрибуты узлов принадлежат разным таблицам. Для каждой пары атрибутов алгоритм определяет кратчайший путь как минимальную сумму длин дуг, соединяющих узлы атрибутов. Программной реализацией алгоритма является функция *short_path*(*attr_name_begin, attr_name_end*).

Третий этап из всех этапов позволяет максимально сократить трудоемкость процесса УД. Но его эффективность будет зависеть от наличия в организации расширенной структурно-должностной иерархии сотрудников.

Четвертый этап автоматически создает правила УД на основе механизмов СУБД, который использует операции УД, принятые в конкретной СУБД. Для формализации этого процесса предложено представлять все операции разных СУБД в виде шаблона операций программного кода, который учитывает разные типы СУБД, разные политики безопасности, разные алгоритмы реализации политик безопасности. Предложено описывать шаблон в виде кортежа < *dbms, policy, alg_type, pattern* >,
где *dbms* – тип СУБД (*Oracle, IBM DB2, MS SQL, Sybase, PostgreSQL*);

policy = {*MAC, DAC*} – тип используемой политики безопасности при УД;

alg_type = {*Native, Schema*} – тип алгоритма реализации используемой политики безопасности, который может использовать встроенные в СУБД операции УД (*Native*) или основанные на механизме схем данных и виртуальных таблиц пользователя (*Schema*);

pattern – шаблон программного кода при реализации алгоритма УД.

Предложенная унификация процесса УД позволяет уменьшить различия в операциях описания УД для разных СУБД.

Реализация политики УД при *alg_type = 'Schema'* поддерживается СУБД, в которой существуют операции создания схем пользовательских данных (*create/alter schema*) и операции создания виртуальных таблиц (*create view*), которые могут быть обновляемыми через встроенные механизмы или через использование дополнительных программных механизмов активных СУБД ('*instead of*' *triggers* или *roles*). Далее предложены примеры описания шаблонов программного кода при реализации алгоритма УД для *alg_type = 'Native'* для СУБД *Oracle, IBM DB2 и Sybase*, а также для *alg_type = 'Schema'* с *policy = 'DAC'*.

В примерах присутствуют переменные: *$user_name, $table_name, $column_name* – имена пользователя, таблицы и первичного атрибута таблицы, соответственно, которые участвуют в политике типа *DAC, $user_view* – описание представления пользователя-субъекта о таблицах-объектах в виде *SQL*-запроса, ограничивающего доступ к данным, а также *$label_list* – список уровней доступа и меток конфиденциальности для политики типа *MAC*. Значение переменной *$user_view* определяется на третьем этапе.

На рис. 2 представлен шаблон программного кода при реализации алгоритма УД с *alg_type = 'Native', policy = 'DAC'* СУБД *Oracle*.

```
Create function access_rule_$table_name_$column_name (p_schema  in varchar2, p_object in varchar2 )
return varchar2 as begin return '$column_name in ( $user_view)';  end;  /
begin dbms_rls.add_policy(object_schema => ‘$user_name',
object_name =>'$table_name',
policy_name =>'$table_name_$column_name',
function_schema => '$user_name',
policy_function => ' access_rule_$table_name_$column_name',
statement_types => 'select, insert, update, delete',
update_check => FALSE, enable => TRUE,
static_policy => FALSE); end; /
```

Рис. 2. Шаблон программного кода при реализации алгоритма для СУБД *Oracle*

На рис. 3 представлен шаблон программного кода при реализации алгоритма УД с *alg_type = 'Schema', policy = 'DAC'* СУБД *PostgreSQL*.

```
Create schema $user_name;
Create view $user_name.$table_name as
select * from $table_name where $column_name in ( $user_view);
revoke all on $table_name from $user_name;
grant all on $shema_name.$table_name to $user_name;
```

Рис. 3. Шаблон программного кода при реализации алгоритма для СУБД *PostgreSQL*

Для демонстрации работы предложенных алгоритмов рассмотрим БД ИС под управлением СУБД *PostgreSQL*, которая включает таблицы: Persons (pn-id, name), Institutes (in-id, name), Posts (pt-id, name), Staff (pn-id, pt-id, in-id), Courses (cr-id, name), EducationPlan (pn-id, cr-id), Students (pn-id, in-id), StudEducation (pn-id, cr-id).

Пример заполнения БД представлен на рис. 4, а.

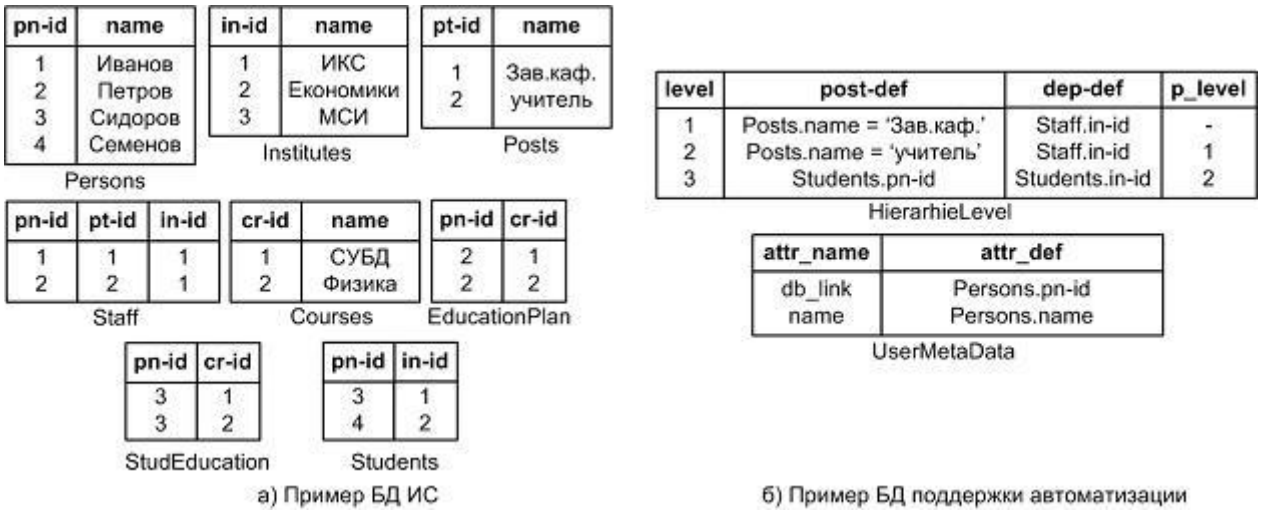

Рис. 4. Пример заполнения таблиц БД: а – пример БД ИС; б – пример поддержки автоматизации

Для автоматического создания правил УД администратору БД необходимо описать таблицы с метаданными по учетным записям субъектов и структурно-должностной иерархии. Примеры заполнения таблиц представлены на рис. 4, б.

В результате работы алгоритмов на первом этапе автоматически будут сформированы учетные записи пользователей ИС: *ivanov, petrov, sidorov, semenov*.

В результате работы алгоритмов на втором этапе для каждого пользователя будет автоматически установлен его уровень доступа к БД: *ivanov* – 1-й уровень, *petrov* – 2-й уровень, *sidorov* и semenov – 3-й уровень.

В результате работы алгоритмов третьего и четвертого этапов будут автоматически созданы правила УД, примеры которых представлены на рис. 5.

Третий этап формирует представление пользователей *ivanov* и *petrov* о содержимом таблицы *persons.* Четвертый этап оформляет созданные представления пользователей с учетом операций управления СУБД *PostgeSQL*.

```
create schema ivanov;
create view ivanov.persons as select * from persons where pn-id in ((select persons.pn-id from persons p where p.name
= 'ivanov')
union
(select pn-id from staff s1 where exists ( select s2.* from persons p, staff s2 where p.name = 'ivanov' and s2.in-id =
s1.in-id and s2.pn-id = p.pn-id)
union
(select pn-id from students s1 where exists ( select s2.* from persons p, staff s2 where p.name = 'ivanov' and s2.in-id =
s1.in-id and s2.pn-id = p.pn-id));
create schema petrov;
create view petrov.persons as select * from persons where pn-id in ((select persons.pn-id from persons where name =
'petrov')
union
(select pn-id from Students s, Education e1 where s.pn-id = e.pn-id and exists ( select s2.* from persons p,
EducationPlan e2 where p.name = 'petrov' and e2.pn-id = p.pn-id and e2.cr-id = e1.cr-id)
```

Рис. 5. Пример правил УД в виде операций СУБД *PostgreSQL*

Как видно из рис. 5, пользователь *ivanov* как заведующий кафедрой будет получать доступ к данным в БД о самом себе, о сотрудниках, которые являются преподавателями с его кафедры, и о студентах, которые обучаются на кафедре. Пользователь *petrov*, являясьпреподавателем, будет получать доступ к данным в БД о самом себе и о студентах, которых он обучает.

Заключение

Предложенная формализация правил управления доступом на основе шаблонов позволила автоматизировать процесс администрирования в ИС, с которой работают пользователи крупных (корпоративных) организаций, и в которой используются СУБД разных типов. Эффективность процесса автоматизации напрямую зависит от существующей в организации структурно-должностной иерархии пользователей. Администратору необходимо лишь заполнить БД иерархии и метаданных получения данных о пользователях из БД ИС, а описания правил доступа к таблицам формируются автоматически.

Дальнейшее развитие разработанной системы предполагает создание метода выбора оптимальной политики безопасности (*DAC*, *MAC*), алгоритма управления (*Native*, *Schema*) с учетом областей функционирования ИС и типов СУБД с многокритериальной оптимизацией по критериям минимального времени выполнения запросов к данным в БД и минимальной трудоемкости процесса планирования политики управления доступом. Важным развитием системы также является разработка алгоритма тестирования качества разработанных правил с учетом установленной политики безопасности, который будет гарантировать отсутствие ошибок администратора БД в процессе управления доступом к данным.

1. *James Joshi, Walid G. Aref, Arif Ghafoor, Eugene H.* Spafford: Security models for web-based applications // Communications of the ACM (CACM), 2001. – 44. – P. 38–44.
2. *Ravi S. Sandhu, David F. Ferraiolo, D. Richard Kuhn:* The NIST model for role-based access control: towards a unified standard. The Procs. of the fifth ACM workshop on Role-based access control, July 26–27, 2000. – Berlin, Germany. – P. 47–63.
3. *Arnon Rosenthal, William R.* Herndon: Granularity of Data Protection for MLS Applications and DBMSs. The Procs. of the IFIP WG11.3 Working Conference on Database Security, Lake Guntersville, Alabama, USA, 12–15 September, 1993. – P. 161–180.
4. *Row-level Security in A. Relational* Database Management System. Patent N 7,240,046 B2. 2007. United States Patent.
5. *Steve Barker:* Dynamic Meta-level Access Control in SQL. The Procs. of 22nd Annual IFIP WG11.3 // Working Conference on Data and Applications Security, London, UK, July 13–16. – 2008. – P. 1–16.
6. *Shariq Rizvi, Alberto Mendelzon, S. Sudarshan, Prasan Roy:* Extending query rewriting techniques for fine-grained access control. The Procs. of the 2004 ACM SIGMOD international conference on Management of data, June 13–18, 2004. – Paris, France. – P. 551-562.
7. *Qihua Wang, Ting Yu, Ninghui Li, Jorge Lobo, Elisa Bertino, Keith Irwin, Ji-Won Byun:* On the correctness criteria of fine-grained access control in relational databases. Proceedings of the 33rd international conference on Very large data bases, September 23–28, 2007. – Vienna, Austria. – P. 555–566.
8. *Steven Dawson, Shelly Qian, Pierangela Samarati:* Providing Security and Interoperation of Heterogeneous Systems. Distributed and Parallel Databases. – 2000. – 8, N 1, January. – P. 119–145.
9. *Torsten Priebe, Eduardo B. Fernández, Jens Ingo Mehlau, Günther Pernul:* A Pattern System for Access Control. // In: Research Directions in Data and Applications Security XVIII, IFIP TC11/WG 11.3 Eighteenth Annual Conference on Data and Applications Security, July 25–28, 2004, Sitges, Catalonia, Spain. – P. 235–249.
10. *Jonathan D. Moffett, Emil Lupu:* The Uses of Role Hierarchies in Access Control. In: Procs. of the Fourth ACM Workshop on Role-Based Access Control, October 28–29, 1999, Fairfax, VA, USA. – P. 153–160.
11. *Sejong Oh, Ravi S. Sandhu, Xinwen Zhang.* An effective role administration model using organization structure. ACM Transactions on Information and System Security (TISSEC). – 2006. – 9. – 2006. – P. 113–137.