УДК 004.624

**И.Сауд**

## ТЕСТИРОВАНИЕ СИСТЕМЫ АВТОМАТИЗИРОВАННОГО КОНТРОЛЯ ДОСТУПА К РЕЛЯЦИОННОЙ БАЗЕ ДАННЫХ

*Посвящена вопросам проверки корректности работы механизма автоматизированного контроля доступа к реляционной базе данных пользователей программ корпоративной информационной системы и управления доступом. Предложена модификация метода разбиения тестовых наборов на классы эквивалентности, которые учитывают SQL-запросы из журналов транзакций и роли пользователей в работе информационной системы. В основе модификации метода используется алгоритм генерации SQL-запросов, работа которого проверена на примере системы электронного деканата университета.*

***Ключевые слова:*** реляционные базы данных, *система контроля доступом, тестирование.*

**E.Saoud**

## TESTING OF RELATIONAL DATABASES AUTOMATIC ACCESS CONTROL SYTEM

*The article is devoted to the validation of automated control access to the relational database of software users in corporate information system. A modification of the method of test case partitioning into equivalence classes, which use SQL-queries from the transaction log and the roles of users in information system, is proposed. The modification of method is based on using of algorithm for SQL-queries generation and there is checked on the E-Dean of University.*

***Keywords:*** *relational databases, access control, testing.*

**І.Сауд**

## ТЕСТУВАННЯ СИСТЕМИ АВТОМАТИЗОВАНОГО КОНТРОЛЮ ДОСТУПА ДО РЕЛЯЦІЙНОЇ БАЗИ ДАНИХ

*Присвячено питанням перевірки коректності роботи механізму автоматизованого контролю доступу до реляційної бази даних користувачів програм корпоративної інформаційної системи і управління доступом. Запропоновано модифікацію методу розбиття тестових наборів на класи еквівалентності, які враховують SQL-запити з журналів транзакцій і ролі користувачів в роботі інформаційної системи. В основі модифікації методу використовується алгоритм генерації SQL-запитів, робота якого перевірена на прикладі системи електронного деканату університету.*

***Ключові слова****: реляційні бази даних, система контролю доступом, тестування.*

**Введение.** Механизм контроля доступа пользователей к таблицам реляционной базы данных (БД) корпоративной информационной системы (ИС) должен разрешать/запрещать читать/редактировать данные в соответствии с установленными в организации должностными инструкциями. Эти требования являются необходимым условием для удовлетворения требований раздела «Контроль доступа» из международного стандарта по информационной безопасности ISO 17799-2005 или его украинской редакции ДСТУ ISO/IEC 27001:2010.

Для администратора ИС ручной процесс настройки механизма контроля доступа в ИС с сотнями пользователей и сотнями таблиц БД сопряжен с ошибками, особенно при использовании механизма горизонтальной и вертикальной детализации таблиц на основе их виртуализации [6]. Разработка автоматизированной системы контроля доступа к БД (САУД) и управления этими правами позволяет сократить трудоемкость и вероятность возникновения ошибок [4, 7]. В тоже время, как отмечается в стандартах, для администратора ИС необходима гарантия корректной работы САУД. Доказательство корректной работы можно получить на основе двух способов: формализации данных и алгоритмов их обработки в процессе моделирования работы САУД [2, 3], а также на основе тестирования программных модулей САУД [1, 5]. Достоинством второго способа является его включение в современную методологию разработки программных систем и адаптация к жизненному циклу систем обеспечения защиты данных.

Правильность любой системы может быть проверена на основе изучения программного кода (*White-Box-testing*) или спецификаций (*Black-Box-testing*) [1], которые

должны уменьшить количество необходимых тестов и длительность тестирования. Для программ доступа к БД из-за декларативности языка *SQL*–запросов и необходимости учета состояния самой БД чаще используется вторая группа способов [5]: эквивалентное разбиение, анализ граничных значений, анализ причинно–следственных связей.

Выбор способа тестирования определяется многообразием задач тестирования БД для контроля: ограничений целостности таблиц БД, корректности *SQL*–запросов в программном коде, корректности правил активных БД. В то же время в работах, посвященных механизму управления доступом к таблицам на уровне представлений пользователей, отсутствуют примеры применения тестирования для проверки корректности работы названого механизма. Поэтому целью работы является сокращение множества тестовых наборов, обеспечивающих проверку корректности работы САУД.

**Концепция тестирования САУД.** Любой процесс тестирования методами *Black-box* включает четыре этапа [1]: определение спецификации, генерация тестов, выполнение тестов и оценка результатов тестирования. На рис. 1 представлена схема взаимодействия компонент системы тестирования работы САУД. Структура упомянутых этапов разработана с учетом особенностей работы САУД.

В этап определения спецификаций включены три шага: *1)* получение файла журнала *SQL*–запросов транзакций, выполняемых программами ИС, и сохраняемых внутренними средствами работы СУБД; *2)* автоматический перенос *SQL*–запросов из файла в БД с одновременным их преобразованием в XML–деревья; *3)* получение таблиц со структурно-должностной иерархией и информацией по авторизации пользователей ИС.

Этап генерации тестовых запросов состоит из четырех шагов: *1)* получения из БД *SQL*–запросов и их *XML*-деревьев; *2)* получения по каждому пользователю структурно-должностной иерархии; *3)* для каждого *SQL*–запроса генерации тестовых запросов с учетом каждого пользователя; *4)* разделения тестовых запросов на корректные и некорректные с учетом пользователей, которые их будут выполнять, и сохранение тестов БД.

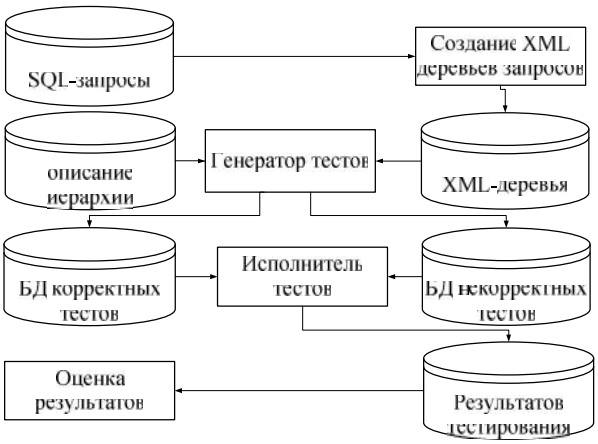


Рис. 1. Схема взаимодействия компонент системы тестирования

Разработка тестов методом эквивалентного разбиения осуществляется в два этапа: выделение классов эквивалентности (в дальнейшем, классов); построение тестов. Классы выделяются путем выбора каждого входного условия и разбиения его на две или более групп. При этом различают два типа классов: правильные, представляющие правильные входные данные программы, и неправильные – ошибочные входные значения.

Предлагается оформлять классы на основе *SQL*–запросов журнала транзакций, в которых правильные значения констант из *SQL*–запроса определяют правильные классы относительно определенного пользователя, а не правильные значения констант *SQL*-запроса определяют неправильные классы.

Процесс построения тестов включает три шага. *1)* назначение каждому классу уникального номера; *2)* проектирование новых тестов, каждый из которых покрывает как можно большее число не покрытых правильных классов, до тех пор, пока все правильные классы не будут покрыты тестами; *3)* запись тестов, каждый из которых покрывает один из непокрытых неправильных классов, пока все неправильные классы не будут покрыты тестами.

В этап исполнения тестовых запросов предложено включать такие шаги: *1)* получение тестовых запросов из БД корректных тестов и некорректных тестов; *2)* получение из таблиц описания иерархии и описания

пользователей информации о проверяемом пользователе; *3)* выполнение тестовых запросов от имени определенного пользователя; *4)* сохранение результатов выполненных тестов в БД результатов тестирования.

Этап оценки результатов тестовых запросов состоит из следующих шагов. *1)* получения тестовых запросов из БД корректных и некорректных тестов; *2)* получения результатов тестирования из БД; *3)* оценки результатов на основании анализа пустых ответов в тестах и определение корректных ответов. Корректный ответ: непустой ответ для правильных тестов и пустой – для неправильных.

**Структуры данных обеспечения процесса тестирования.** Совокупность запросов файла журналирования представим как *Q = {< q, xmlt >}*, где *q – SQL*–запрос транзакций, получаемый внутренними средствами работы СУБД; *xmlt – XML*–дерево запроса. Множество констант *where*–фразы *XML*-дерева запроса представим в виде *Const = {< rn, an, oper, const >}*, где *rn* – таблица; *an* – атрибут; *oper* – операции связи между атрибутами; *const* – константы. Множество корректных и некорректных тестовых запросов представим как *CORT={< stq, cq >}* и *INCORT={< stq, incq >}*, соответственно, где *stq* – ссылка на таблицу описания пользователей ИС; *cq, incq* – корректные и некорректные тестовые запросы.

Множество корректных и некорректных результатов исполняемых корректных запросов представим как *CRT={<crt>}* и *INCRT={<incrt>}*, соответственно, где *crt, incrt* – результат тестирования исполняемого корректного и некорректного тестового запроса.

Структурно–должностную иерархию в организации представим как множество *OH={< hn, hd, dd >}* , где *hn* – имя уровня иерархии; *hd* – атрибуты описания должности по иерархии в БД; *dd* – атрибуты описания структуры по иерархии в БД. Описание пользователей ИС представим как множество *UN={< un , hn >}*, где *un*–логин пользователя при доступе к БД ИС.

**Алгоритм генерации тестов.** Число тестовых наборов, которые необходимо подготовить методом эквивалентного разбиения, можно определить по формуле $|Q|*|UN|(1+|U|)$, где $U \subset UN \wedge U := \{u_i\} \wedge u_i =< un_i, hd_i > \wedge hd_i = hd_C \wedge un_i \neq un_C$

Процесс ручной подготовки КЭ может быть автоматизирован, если: 1) администратор подготовит описание структурно-должностной иерархии; 2) ожидаемые результаты тестов можно представить в виде пустого или непустого ответа на *SQL*–запрос теста.

Входными параметрами работы алгоритма являются множество номеров личности *PID*, множество имен пользователей *UN*, множество должностной иерархии *HD*, множество структурной иерархии *DD*, множество *XML*–деревьев запросов *XMLT*. На выходе работы алгоритма получаем множество тестовых запросов *CORT, INCORT*.

Работа алгоритма представлена следующими действиями с использованием формализаций реляционной алгебры и логики предикатов:

$$\text{for } un_i \in UN \text{ do}$$
$$xmlt_i = \{< const_i, lo_j, link_k >\} : const_i \in CONST$$
$$\wedge lo_j \in LO \wedge link_k \in LINK$$
$$const_i :=< c_i, op_j, n_k >: c_i \in C \wedge op_j \in OP \wedge$$
$$n_k \in N, CONST := CONST \cup \{const_i\}$$
$$CSS := const_1, ..., const_n : const_n \in CONST$$
$$WS := f(t_i.a_i, t_j.a_j) : t_i, t_j \in T \wedge a_i, a_j \in A$$
$$FS := ffs(ws),\ COR := \{\prod_{css}(><_{ws})(fs)\}$$
$$SS := cor_1, ..., cor_n : cor_n \in COR$$
$$CORT := \{\prod_{ss}(><_{ws})(fs)\}$$
$$INCOR := \{\prod_{css}(><_{ws})(fs)\}$$
$$ISS := incor_1, ..., incor_n : incor_n \in INCOR$$
$$INCORT := \{\prod_{iss}(><_{ws})(fs)\}$$
$$return\{CORT, INCORT\}$$

**Выполнение тестов.** Входными параметрами работы алгоритма являются множества правильных тестов *CORT*, неправильных тестов *INCORT* и множество пользователей *UN*.

На выходе работы алгоритма получаем множества *CRT, INCR*. Работа алгоритма представлена такими действиями с исполь-

зованием формализаций реляционной алгебры и логики предикатов:

for $un_i \in UN$ do

connect $(dp, un_i) : \forall un_i \in UN$

if $cort_i = \phi \, \forall cort_i \in CORT$ then

$RCT_i := RCT \cup \{rct_i, 'F'\}, CRT := \{RCT_i\}$

else $RCT_i := RCT \cup \{rct_i, 'T'\}$

$CRT := CRT \cup \{RCT_i\}$ endif

*if* $incort_i \neq \phi \, \forall \, incort_i \in INCORT$ *then*

$RINCT_i := RINCT \cup \{rinct_i, 'T'\}$

$INCRT := INCRT \cup \{RINCT_i\}$

*else*$RINCT_i := RINCT \cup \{rinct_i, 'F'\}$

$INCRT := INCRT \cup \{RINCT_i\}$

*endif*

*return*$\{CRT, INCRT\}$.

**Оценка результатов тестирования**. Входными параметрами работы алгоритма являются множества результатов выполнения правильных тестов *CRT*, результатов выполнения неправильных тестов *INCRT*, и множество результатов выполнения всех тестов $RT:=\{CRT\} \cup \{INCRT\}$, На выходе работы алгоритма получаем множество некорректных результатов тестирования *INCFT*.

Работа алгоритма представлена такими действиями с использованием формализаций реляционной алгебры и логики предикатов:

for $rt_i \in RT$ do

if $rt_i := 'F' : rt_i \in CRT \wedge rt_i \in RT$ then

$CCRTF := \{rt_i\}$ else $CCRTT := \{rt_i\}$

endif;

if $rt_i := 'F' : rt_i \in INCRT \wedge rt_i \in RT$ then

$CINCRTT := \{rt_i\}$ else $CINCRTT := \{rt_i\}$

endif;

$CFT := \{CCRTF\} \cup \{CINCRTT\}$

return{CFT}

**Алгоритм формирования графа.** Входной параметр работы алгоритма – множество *XML*–деревьев запросов. На выходе работы алгоритма получаем множество связанных узлов $G_i$ . Работа алгоритма представляется следующими действиями с использованием формализаций реляционной алгебры и логики предикатов:

for $ws_i \in XMLT$ do

$ws_i := \{< t_i.a_i, op_j, t_k.a_k >: t_i, t_k \in T\}$

$n_i := t_i.a_i : n_i \in N_i, n_i := i : i = 1,...,r$

$n_k := t_k.a_k : n_k \in N_k, n_k := k : k = 1,...r$

*if* $(op_j = '=') or (t_i = t_k)$*then*

$n_i = n_k, G_i := G_i \cup \{n_i, n_j\}$

*endif*

*return*$\{G_i\}$.

Построенный граф используется как временная структура для работы функции *f(n1,n2)* по определению пути между атрибутами таблиц *FROM*–фразы запроса из алгоритма генерации тестов. При поиске путей используется алгоритм кратчайшего пути Дейкстры.

**Апробация работы алгоритмов тестирования.** Проверка работы предложенных алгоритмов выполнена на примере ИС «Электронный деканат». Примером SQL-запроса может быть:

*Select t1.a1,t4.a3 from t1,t2 where t1.a1=t2.a2 and*

*t3.a4=t4.a3.*

На рис. 2 представлена *where*–фраза указанного запроса в виде *XML*–дерева.

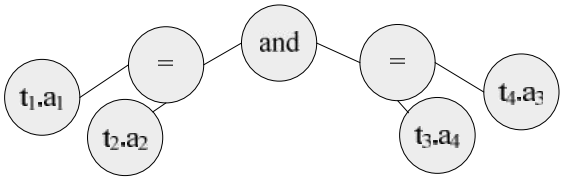


Рис. 2. Пример дерева *where*–фразы

На рис. 3 показан пример графа, полученного в результате работы алгоритма формирования графа.

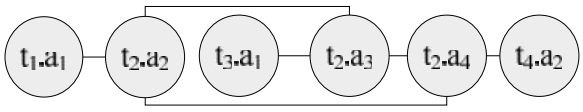


Рис. 3. Пример графа *SQL*–запроса

В таблице приведены примеры КЭ, созданные на основе анализа структурно-должностной иерархии пользователей университета (11 деканов, 6 зав. кафедрами одного из факультетов). В скобках для каждого класса указано количество тестов.

1. Примеры КЭ

| Входные условия | Правильные КЭ | Неправильные КЭ |
|---|---|---|
| Декан факультета 1 | (1) Студент факультета | (10) Студенты других факультетов |
| Заф.кафедрой 1 | (1) Студент кафедры1 | (5) Студенты других кафедр факультета |

При использовании метода простого перебора число тестов определяется количеством студентов, которые учатся на факультетах и кафедрах, что указывает на сокращение тестов в тысячи и сотни раз соответственно.

**Выводы.** Предложенный метод тестирования на основе классов эквивалентности позволяет проверить корректность работы программных модулей САУД при условии наличия журнала *SQL*–запросов транзакций, сформированного в результате работы программ ИС с БД. При этом число необходимых тестовых наборов, подготавливаемых администратором вручную, сокращается многократно по сравнению с методом простого перебора. Исключение ручной работы стало возможным при учете структурно-должностной иерархии пользователей и описаний правил предоставления прав доступа пользователей из должностных инструкций сотрудников. В то же время правильность указанных данных может быть подтверждена только формальными способами доказательств, что указывает на необходимость дальнейшей разработки системы, объединяющей методы формальной верификации спецификаций и методы тестирования программных модулей, работающих по заданным спецификациям.

Список использованной литературы

1. Бейзер Б. Тестирование черного ящика. Технологии функционального тестирования программного обеспечения и систем. – СПб: Питер, 2004. – 320 с.

2. Тарасов Д.О. Формальні моделі систем захисту інформації реляційних баз даних. / Д.О. Тарасов // Інформаційні системи та мережі. Вісник НУ “Львівська політехніка”. – 2003. – № 489. – С. 296–306.

3. Сауд І. Формальна специфікація моделі управлення доступом в реляційних базах даних з використанням мови UML та OCL // Праці V міжнародної наукової конференції молодих вчених "Комп'ютерні науки та інженерія 2011", 24-26 Листопада, 2011, Львів: Україна. – С. 68–69.

4. Blazhko A.A., Antoshchuk S.G., Saoud E. Automated Design Method of Hierarchical Access Control In Database // The Procs. of 5th IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, September 21–23, 2009, Rende (Cosenza), Italy. – P. 361–363.

5. Chan M.Y., Cheung S.C., Testing Database Applications with SQL Semantics // The Procs. of 2nd International Symposium on Cooperative Database Sys-tems for Advanced Applications (CODAS'99), Wollon-gong, Australia, March 1999. – P. 363–374.

6. Row-level Security in A Relational Database Management System. Patent N 7,240,046 B2. 2007. United States Patent.

7. Wang Q., Yu T., Li N., Lobo J., Bertino E., Irwin K., Byun J. On the correctness criteria of fine-grained access control in relational databases // The Procs. of the 33rd international conference on Very large data bases, September 23-28, 2007, Vienna, Austria. – P. 555–566.

Получено 13.02.2012

References

1. Beizer B. Black-Box Testing: Techniques for Functional Testing of Software and Systems. – Wiley: New York, 1995. – 320 p. [in English].

2. Tarasov D. Formal models of protection of relational databases / Information Systems and Networks. Bulletin "Lviv Polytechnic". – 2003. – № 489. – P. 296–306 [in Ukrainian].

3. Saoud E. Formal Specification of Access Control Model for Relational Databases using UML with OCL // In the Procs. of 5th International Conference of Young Scientists “Com

puter Science & Engineering 2011” (CSE-2011), November 24-26, 2011, Lviv, Ukraine. – P. 68–69 [in Ukrainian].

4. Blazhko A.A., Antoshchuk S.G., Saoud E. Automated Design Method of Hierarchical Access Control In Database / The Procs. of 5th IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, September 21–23, 2009, Rende (Cosenza), Italy. – P. 361–363 [in English].

5. Chan M.Y., Cheung S.C., Testing Database Applications with SQL Semantics // The Procs. of 2nd International Symposium on Cooperative Database Sys-tems for Advanced Applications (CODAS'99), Wollon-gong, Australia, March 1999. – P. 363–374 [in English].

6. Row-level Security in A Relational Database Management System. Patent N 7,240,046 B2. 2007. United States Patent [in English].

7. Wang Q., Yu T., Li N., Lobo J., Bertino E., Irwin K., Byun J. On the correctness criteria of fine-grained access control in relational databases // The Procs. of the 33rd international conference on Very large data bases, September 23-28, 2007, Vienna, Austria. – P. 555–566 [in English].

Сауд Ибаа,
аспирант каф. системного программного обеспечения Одесского нац. политехн. ун-та,
тел. 8048734566.
E-mail: ebaa005@mail.ru