УДК 004.624

А. А. Блажко, канд. техн. наук,
И. Сауд

ОЦЕНКА ПРОИЗВОДИТЕЛЬНОСТИ РАБОТЫ МЕХАНИЗМА РАЗГРАНИЧЕНИЯ ДОСТУПА ПОЛЬЗОВАТЕЛЕЙ К БАЗАМ ДАННЫХ ИНФОРМАЦИОННЫХ СИСТЕМ

Аннотация. *Посвящена вопросам проверки свойств масштабируемости механизмов управления доступом к данным. Выполнены экспериментальные оценки эффективности используемых операций в запросах ограничения доступа, продолжительности задержки операций для базы данных разных объемов, продолжительности задержки операций для разных типов механизмов, а также оценки нагрузочного тестирования для базы данных разных классов информационных систем.*

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

A. A. Blazhko, PhD.,
E. Saoud

PERFORMANCE EVALUATION OF THE ACCESS CONTROL MECHANISM FOR USERS IN DATABASES OF INFORMATION SYSTEMS

Abstract. *The article is devoted to testing the scalability properties of the mechanisms of data access control. The experimental evaluation of the effectiveness of operations used in queries, limited access, evaluation of delay operations for a database of different volumes, estimates the delay of operations for different types of assessment tools and databases for load testing of different classes of information systems.*

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

О.А. Блажко, канд. техн. наук,
Сауд И.

ОЦІНКА ПРОДУКТИВНОСТІ РОБОТИ МЕХАНІЗМУ РОЗМЕЖУВАННЯ ДОСТУПУ КОРИСТУВАЧІВ ДО БАЗ ДАНИХ ІНФОРМАЦІЙНИХ СИСТЕМ

Анотація *Присвячено питанням перевірки властивостей масштабованості механізмів управління доступом до даних. Виконано експериментальні оцінки ефективності використовуваних операцій в запитах обмеження доступу,тривалості затримки операцій для бази даних різних об'ємів, тривалості затримки операцій для різних типів механізмів, а також оцінки навантажувального тестування для бази даних різних класів інформаційних систем.*

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

Введение. Разграничение доступа пользователей к таблицам реляционной базы данных (БД) информационной системы (ИС) может реализовываться на двух вариантах уровней программного кода: уровень прикладной программы и уровень системного модуля системы управления базами данных (СУБД). На рис. 1 представлена концепция работы на этих уровнях и основные их недостатки. Первый вариант предполагает включение правил доступа к таблицам БД в программный код программы–клиента.

Второй вариант использует программы-сервера СУБД с механизмом активных БД [1] и в основном именуется как *Row Level Security* (*RLS*) или *Virtual Private Database* (*VPD*). У каждой СУБД существуют особенности работы механизма активных БД. В СУБД *Oracle* применяются виртуальные таблицы и *instead-of*-триггера их обновления.

Система управления базой данных *PostgreSQL* для обновления виртуальных таблиц предусмотрен механизм правил. В СУБД также могут использоваться встроенные системы разграничения доступом, например как пакет *dbms_rls* в СУБД *Oracle.*

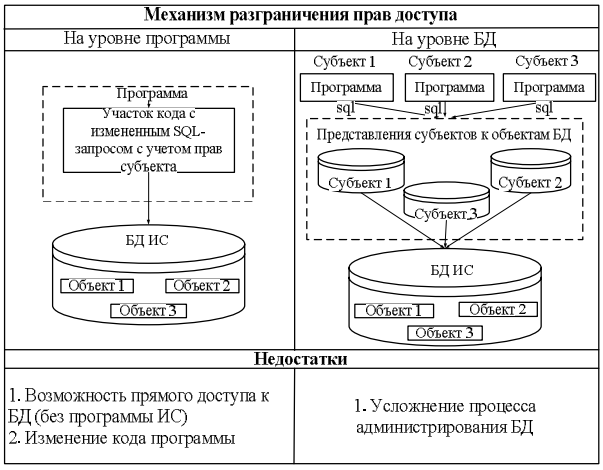


Рис. 1. Концепция способов управления доступом

Отмеченные недостатки *RLS*-механизма были решены при использовании системы автоматизированного управления доступом (САУД) [2], корректность работы которой подтверждена функциональным тестированием [3]. В то же время как дополнительный программный модуль *RLS*–механизм вносит дополнительную задержку в длительность выполнения *SQL*–запросов к БД. Даже когда безопасность системы важнее производительности, в процессе ее внедрения необходимо оценить масштабируемость в виде линейной зависимости показателей производительности, например, времени выполнения запросов, от количества пользователей и объемов БД.

Такая оценка должна быть выполнена с учетом особенностей использования операторов *SQL*–запросов для разных СУБД, разных типов связей между таблицами БД, разных типов *RLS*–механизмов и разных типов транзакций в классификации предметных областей ИС. Но это возможно только на основе натурных экспериментов, результаты которых не публиковались специалистами в области управления доступом к БД.

Поэтому целью данной работы является экспериментальное подтверждение масштабируемости работы *RLS*–механизма на основе оценки эффективности используемых операций в *SQL*–запросах ограничения доступа, оценки задержки операций для БД разных объемов, оценки задержки операций для разных типов *RLS*–механизма и оценки нагрузочного тестирования для БД разных классов ИС.

Оценка временной задержки операций для разных типов RLS–механизма. Система САУД использует автоматизацию на основе шаблонных структур данных, представленных в виде пятерки

< *dbms, policy, alg_type, oper, pattern* >,

где *dbms* – тип СУБД (Oracle, IBM DB2, MS SQL, Sybase, PostgreSQL, UNI); *policy* = {MAC, DAC} – тип политики управления доступом; *alg_type* = {*Native, Schema*} – алгоритм реализации политики, который использует встроенные команды по управлению доступом (*Native*) или основан на пользовательских схемах данных (*Schema*); *oper* – тип операции доступа к таблице (*select, insert, update, delete*); *pattern* – шаблон команд по управлению доступом.

Унификация процесса управления доступом предполагает сокращение различий в описании команд для разных СУБД. При совпадении последовательности команд для разных СУБД можно их выделить в отдельный *UNI*-тип СУБД.

Реализацию политики управления доступом на основе пользовательских схем данных могут поддерживать СУБД, в которых присутствуют команды управления схемами (*create/alter schema*) и команды управления виртуальными таблицами (*create view*), обновляемые за счет использования встроенных механизмов обновления или за счет программирования с использованием триггеров типа *instead of*.

Для шаблонной структуры типа <*Oracle, DAC, Native, {select, insert, update, delete}, pattern*> элемент *pattern* может быть представлен как

```
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; /
```

В представленном шаблоне используются переменные: *#user_name#* – имя пользователя, *#table_name#* – имя таблицы, *#column_name#* – имя атрибута таблицы, который участвует в ограничении доступа пользователя к картежам таблицы *#user_view#* – предикат условий ограничений доступа к картежам таблицы.

Политика управления доступом типа *Native* также поддерживается СУБД *IBM DB2* и *Sybase*.

Для шаблонной структуры типа < *PostgreSQL, DAC, Schema, select, pattern*> элемент *pattern* может быть представлен как

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#;

Оценка эффективности используемых операций в SQL-запросах ограничения доступа. При описании предикатов ограничения доступа к таблице можно использовать *два варианта* установления пути между ней и *subject*-таблицей: *Semi-Join*-соединение с оператором *In* или с оператором *Exists*. Например, для связи таблиц *R1*и *R2* можно выполнить два эквивалентных запроса:

1) *Select * from R1 where exists (select 1 from R2 where R2.A2 = R1.A1 and P1);*
2) *Select * from R1 where A1 in (select 1 from R2 where P1);*

Native–команды настройки *RLS*-механизма СУБД *Oracle* предполагают динамическое подключение дополнительных предикатов к основному запросу доступа к данным. Поэтому синтаксис допускает только использование *Semi-Join*-соединения с оператором *IN*. Для *Schema*-команд возможны два варианта.

В процессе оптимизации запросов СУБД могут выполнять преобразование операторов *Exists* и *In*. В то же время в руководстве по настройке производительности *Oracle* отмечается, что время выполнения запроса с оператором *Exists* меньше времени выполнения запроса с оператором *In* при большой селективности предиката *P1*. Селективность зависит от типа связи между таблицами в реляционной модели БД. На рис. 2 представлены два вида модели БД со связями: «один к одному» (*1-1*), «один ко многим» (*1-M*). Естественно, чем больше длина предиката связи между *subjec*-таблицей и таблицами предметной области, тем больше влияние типа операции на общую стоимость выполнения запроса.

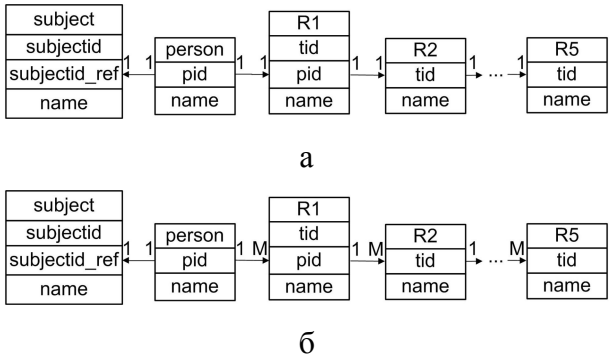


Рис. 2. Модели БД для проведения тестирования: а) модель БД со связью «1-1»; б) модель БД со связью «1-M»

Для экспериментальной оценки эффективности способов соединения проведены эксперименты, включающие серии тестов для пяти вариантов длин предикатов связи $pl \in \{1,2,3,4,5\}$, для трех размеров таблиц БД ($k*10^6$ строк, k = 1,2,3) с использованием *alg_type* = {*Native, Schema}* в СУБД *Oracle* и типами связей *Exists* и *In*. Результаты представлены на рис. 3.

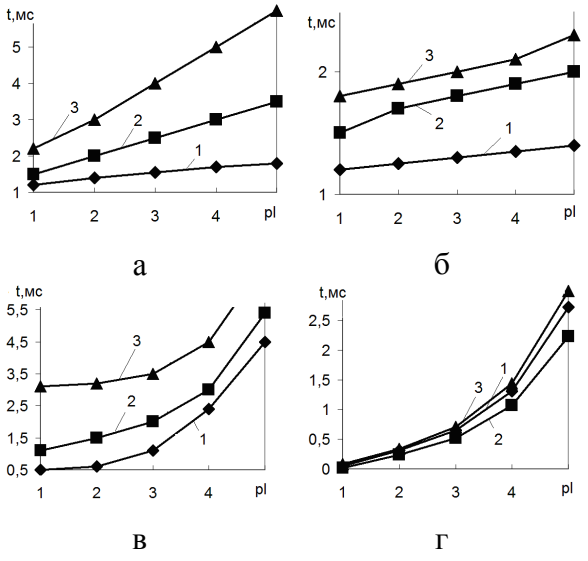


Рис. 3. Оценка времени выполнения операции при 1 –*к* = 1; 2– *к* = 2; 3–*к* = 3:
а – тип связи «1-M», *EXISTS* и *IN*, *select*;
б – тип связи «1-1», *EXISTS* и *IN*, *select*;
в – тип связи «1-1», *Native*, *select*;
г – тип связи «1-1», *Native*, *update*

Результаты экспериментов не показали различий во времени выполнения запросов для операторов *Exists* и *In* при *alg_type = Schema*. Как видно из рисунка, наблюдается линейная зависимость увеличения времени при увеличении длины предикатов связи. При этом для типа связи «1-1» угол наклона прямой не меняется при увеличении размера таблиц БД, а для типа связи «1-М» угол наклона увеличивается на 10-15°. Это указывает на масштабируемость *RLS*-механизма при *alg_type=Schema*. В то же время, *alg_type=Native* показал нелинейную зависимость времени выполнения запроса от длины предикатов, особенно для *select*-операции при увеличении размера таблиц БД, что указывает на неудовлетворительную масштабируемость *RLS*-механизма.

Дополнительная задержка на выполнение *select*-операции к произвольной таблице с учетом *RLS*-механизма определяется только временем установления пути между ней и *subject*-таблицей. Для операций *update* и *delete* задержка эквивалентна, т.к. они выполняются над виртуальными таблицами, определяемыми тем же запросом, что и для *select*-операции.

Нагрузочно-объемное тестирование RLS-механизмов для разных классов ИС.

При нагрузочном тестировании желательно использовать тесты, учитывающие разные классы ИС. Общеизвестными тестами являются *TPC* (*Transaction Processing Performance Council*), например *TPC-B* (ИС класса «простая банковская система»), *TPC-D* (ИС класса «система аналитической обработки поставок») [4]. Тест *TPC-B* имитирует систему ввода заказов, вне зависимости от области деятельности, будь-то система управления предприятием или заказ авиабилетов. Тест *TPC-D* ориентирован на *DSS*-системы и использует 17 аналитических сложных запросов, которые могут использоваться широкопрофильным поставщиком при расчете цен и скидок, минимальных требований заказчика, общего анализа и прогнозирования рынка и управления поставками. На рис. 4 показана реляционная модель БД теста *TPC-B*, в которой между *subject*-таблицей и таблицами БД длина предикатов связи $pl \in \{1, 2, 3\}$.

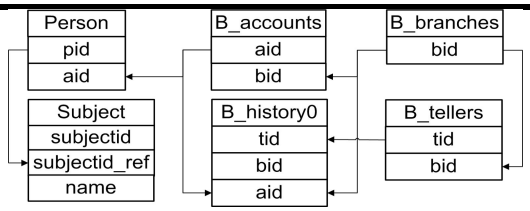
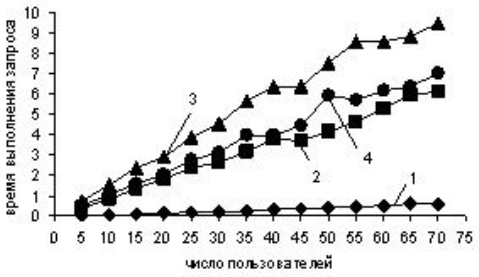


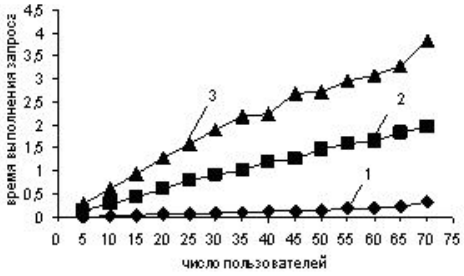
Рис. 4. Реляционная модель БД теста *TPC-B*

С использованием программного обеспечения САУД для теста *TPC-B* были созданы команды работы *RLS*-механизма.

Используя демо-версию системы нагрузочного тестирования *BenchFactory*, провели серию тестов *TPC-B* для СУБД *Oracle*, *PostgreSQL*, результаты которых представлены на рис. 5. В процессе тестирования для разного количества одновременно работающих пользователей (от 5 до 70) система определяла среднее время выполнения запроса в мс.



а



б

Рис. 5. Результаты нагрузочного тестирования тестом *TPC-B:*

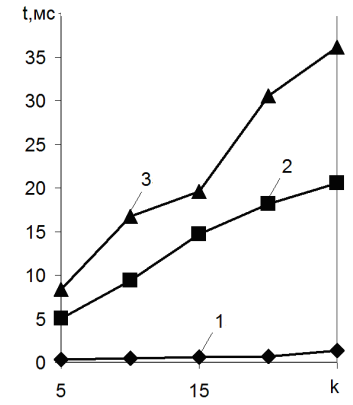
а – СУБД *Oracl* (1–без схемы; 2–схема (EXIST); 3–схема (IN); 4–RLS);
б – СУБД *PostgreSQL Oracle* (1–без схемы; 2–схема (EXIST); 3–схема (IN))

Как видно из рисунков, все зависимости являются линейными, нос разными углами наклона. При *alg_type=Schema* с использованием в запросе оператора *Exists* прямая времени выполнения запросов имеет угол на-

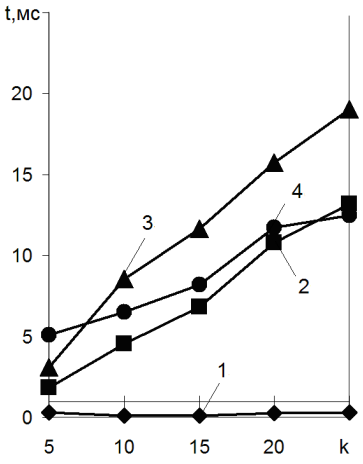
клона на 20° меньше, чем при использовании оператора *In*.

При использовании *alg_type=Native* СУБД *Oracle* (*RLS*) в отличие от предыдущих результатов (рис. 3) уже наблюдается линейная зависимость, что связано с постоянным значением длины предикатов связи.

На рис. 6 представлены результаты объемного тестирования тестом *TPC-B* для СУБД *Oracle*, *PostgreSQL*. Коэффициент объема *k* определяет кратность увеличения количества кортежей в таблицах БД. Как видно из рисунка, кривые зависимостей также линейны. Для СУБД *PostgreSQL* при использовании оператора *Exists* прямая имеет наклон на 15° меньше, чем для оператора *In*. Для СУБД *Oracle* наклон на 10° меньше.



а



б

Рис. 6. Результаты объемного тестирования тестом *TPC-B*:
а – PostgreSQL (1–Full Access; 2–Schema (Exists); 3–Schema (In));
б – Oracle (1–Full Access; 2–Schema (Exists), 3–Schema (In); 4–Native)

На рис. 7 показана реляционная модель БД теста *TPC-D*, в которой между *subject*-таблицей и таблицами БД длина предикатов связи $pl \in \{1, 2, 3, 4, 5\}$.

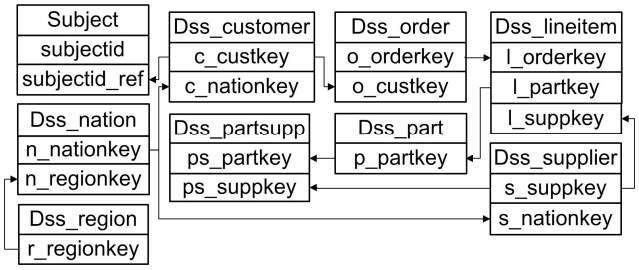


Рис. 7. Реляционная модель БД теста *TPC-D*

Используя программное обеспечение САУД, для теста *TPC-D* создали команды работы *RLS*-механизма. В таблице представлены результаты нагрузочного тестирования тестом *TPC-D* для СУБД *Oracle*, *PostgreSQL* в виде среднего времени выполнения *select*-запросов, измеряемых в секундах.

1. Результаты нагрузочного тестирования тестом *TPC-D*

СУБД	Без схемы	Схема (Exists)	Схема (IN)	Native
Oracle	25.7	7.5	7.3	5,5
PostgreSQL	111	2	5	-

Как видно из таблицы, в СУБД *Oracle* влияние операторов *In* и *Exists* на время выполнения почти эквивалентны, но для СУБД *PostgreSQL* оператор *In* вносит более чем двукратную задержку по сравнению с оператором *Exists*. Но линейность зависимостей указывает масштабируемость *RLS*-механизма для ИС *DSS*-класса.

Выводы

В результате проведенных экспериментов было выявлено, что RLS-механизм с использованием *alg_type=Schema* имеет удовлетворительные свойства по масштабируемости. Если увеличивается длина цепочки предикатов в *select*-запросах, то при использовании *alg_type=Native* масштабируемость теряется. Нагрузочное тестирование для ИС *OLTP*-класса и *DSS*-класса также имеет удовлетворительную масштабируемость для всех типом *RLS*-механизмов. Поэтому система САУД может использоваться в крупных ИС без риска существенных потерь по показателям производительности.

Список использованной литературы

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

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

3. Сауд И. Тестирование системы автоматизированного контроля прав доступа к реляционной базе данных / И. Сауд // Электротехнические и компьютерные системы. – № 05(81). – К.: Техника, 2012. – С. 209–214.

4. Meikel Pöss, Raghunath Othayoth Nambiar, David Walrath Why you should run TPC-DS: a workload analysis // The Procs. of the 33rd international conference on Very large data bases, 2007. – P. – 1138–1149.

Получено 17.06.2012

References

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

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

3. Saud I. Testing of relational databases automatic access control system / I. Saud // Electrotechnic and computer system. – № 05 (81). – Kiev: Technology. – 2012. – P. 209–214 [in English].

4. Meikel Pöss, Raghunath Othayoth Nambiar, David Walrath Why you should run TPC-DS: a workload analysis // The Procs. of the 33rd international conference on Very large data bases, 2007. – P.1138–1149 [in English].

Блажко
Александр Анатольевич,
канд. техн. наук, доц. каф. системного программного обеспечения Одесского нац. политехн. ун-та,
blazhko@ieee.org
тел. 8048734566

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