Indian Journal of Science and Technology, *Vol 9(37), DOI: 10.17485/ijst/2016/v9i37/100393, October 2016*
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Performance Optimization of Row-Level-Data-Security for Transactional Business Intelligence Queries in Cloud and SaaS Business Applications

Arjun K. Sirohi* and Vidushi Sharma

School of ICT, Gautam Buddha University, Yamuna Expressway, Gautam Buddha Nagar, Greater Noida - 201312, Uttar Pradesh, India; asirohi@yahoo.com, vidushi@gbu.ac.in

Abstract

Objectives: To optimize row-level data security for Transactional Business Intelligence (TBI) SQL queries to reduce complexity and enable the back-end database to create better optimized execution plans that perform and scale well. **Methods/Statistical Analysis:** Benchmark experiments were conducted using Oracle RDBMS 11gR2 using representative SQL queries from Oracle's Fusion CRM TBI Applications for five different users with multiple, varying roles and data access permissions. All four measures of SQL performance viz. SQL response time (RT), Input-Output (IO) Buffer Gets, Hard-Parse-Time and Shared Memory utilization were recorded with and without our proposed optimizations. The four performance measures were then compared to record actual improvements. **Findings:** The benchmark experiments established very promising results. We recorded repeatable, significant gains in not only the four measures of individual SQL performance but also at the database resources level. The proposed architecture enables the creation of a hand-shake mechanism between the application and security frameworks to optimize query and application performance. This is done by creating name-value pairs of roles and filter criteria and passing these from the middleware/application to the security framework at run time. The result is a pruned version of the final physical SQL, retaining only relevant RDSPs while removing ones not logically needed. Such optimized row-level data security makes TBI SQL queries less complex and the back-end database is able to create better optimized execution plans that perform and scale well. Query Response Time (RT) improvements ranging from 5% to 745 times, Hard-Parse Time improvements from 1% to 208 times, Logical I/O or Buffer Gets' improvement ranging from 43% to 454 times and SQL-Shared-Memory reduction by up to 52%. **Application/Improvements:** Our proposed architecture is directly applicable to improve performance of all TBI applications that use row-level data security, especially in the Software-as-a-Service (SaaS) and Cloud Models.

Keywords: Access Control and Database Security, Cloud and SaaS Applications, Row-Level Data Security, RBAC, SQL Query Performance, Transactional Business Intelligence

1. Introduction

The recent rise in the use of Software as a Service (SaaS) applications and the shift towards Cloud computing has raised concerns about data security for both the providers and consumers of cloud business applications. The origins of Role Based Access Control (RBAC) can be traced to the paper presented by Ferraiolo and Kuhn at the 15th National Computer Security Conference[1]. Kuhn further elaborated on some aspects of RBAC like mutual exclusion of roles[2]. Later, the proposal by [3]was adopted by the "The American National Standards Institute, International Committee for Information Technology Standards" (ANSI/INCITS) as a standard[4]. Since then, many researchers and practitioners have worked on the many different aspects of RBAC, which reduces the cost of administering access control policies. Chandramouli and Sandhu presented RBAC for commercial database management systems[5]. Chandramouli further described a business process driven framework for RBAC model[6]. In their book titled "RBAC", Ferraiolo,

**Author for correspondence*

Kuhn and Chandramouli have described RBAC and many of its implementation details, including integrating RBAC with Enterprise IT Infrastructures[7]. However, there is only a passing reference to the performance impacts of RBAC in the book, even in the second updated edition[8]. The RBAC standard has since been adopted and implemented by many manufacturers of widely used applications like those from Oracle, Microsoft, IBM, SAP and Sales force. Similarly, the evolution of XACML, the eXtensible Access Control Markup Language, as a standard has led to its adoption across industries as it also supports RBAC. XACML-V3.0, the OASIS Standard for eXtensible Access Control Markup Language (XACML) Version 3.0 has been published and revised a few times[9]. While there has been much research on and adoption of such security implementation standards and their implementation's security benefits, there has not been much research on the impacts of such data security models on the performance and scalability of enterprise business applications in general and specifically on the performance and scalability of transactional business intelligence applications which are resource-intensive even without the effects of row-level data security.

While there are many different platform and application layers involved in data security, this paper examines the existing RBAC approach of applying RDSPs based on the user's job roles, group memberships being part of territory and management hierarchies. In RBAC model, users can be assigned to multiple roles and the data access for such users is the union of all permissions for all the roles. The problems related to access control constraints and user-role assignment was the subject of research published by[10]. The impact of such data security implementations, where users are assigned to multiple roles, on the performance of SQL queries in the underlying relational database is the focus of this paper. While RBAC-based data security may not have huge performance impacts when users have only one defined role, performance regressions start to arise when users are assigned multiple roles and they have access to row-level data via multiple access paths. In such cases, the resulting RDSP sub-queries are the union of all applicable RDSP sub-queries for a given user. For example, a sales manager running a report against a Customer Relationship Management (CRM) application module such as Opportunity Management, could have been granted multiple roles and implicit roles through group memberships and role/territory hierarchies. Based on these multiple roles, the sales manager's access to data is controlled by a union of all RDSPs[11]. Modern enterprise business applications have a very complex data model with the data typically stored in many different database tables. Further, TBI applications involve multiple and complex technologies and are composed of many layers of modelling and abstraction. This results in great complexity in SQL queries' generated at run time. A simple logical request from the application user often results in complex SQL queries. At the back end, the database needs to process these complex queries quickly and send back results to the user in sub-second response time based on Service level Agreements (SLAs) between providers and consumers of such applications. Specifically, SQL queries being generated by TBI applications' query-engines are extremely complex because they involve large scale aggregations, analytics functions and visualizations. Appending of RDSPs to such complex SQL queries often results in poor performance, causing poor user experience and adoption. Kohler and Schaad studied this aspect of user adoption and very aptly observed that "improving the performance of access control decisions will improve the overall performance experienced by the end user significantly"[12].

In this paper, we propose an architecture that limits the performance impacts of applying multiple RDSP sub-queries to the data access SQL queries while still enforcing the required row-level data security. The paper has been divided into five sections. Section 1 is this introduction. Section 2 discusses the current implementations of RBAC and how seeded RDSP sub-queries are used. Section 3 details the impact of applying RDSP on the performance of TBI SQL queries. We highlight the most important limitations in existing architectures. Section 4 describes the contributions of this paper and discusses our proposed architecture to create a framework that acts as a broker between the security framework and application framework resulting in the trimming down of the superset of RDSP sub-queries to only the ones deemed necessary by the application framework before SQL queries are dispatched to the back-end database. The paper then presents results from benchmark experiments conducted to support the proposed architecture in Section 5. The paper concludes with ideas for taking action for implementation, including an actual implementation by Oracle Business Intelligence Enterprise Edition (OBI EE) platform based on the contributions of this paper. The main contributions of this work can be summarized as follows.

- To the best of our knowledge, the proposed architecture is the first specifically designed for Enterprise TBI applications to reduce the complexity of RBAC's RDSPs to improve performance of queries.
- Instead of the standard RBAC implementation which append a union of row level data security predicates in the form of SQL sub-queries, we propose optimizing the row level data security predicates' sub-queries before the final query is sent to the backend database for execution.
- Our proposed architecture of creating a framework for name-value pairs between the enterprise duty roles and the application filter criteria in order to use these pairings at run time to trim the union-all of RDSPs is unique and new.

2. Current State of Row-Level Data Security Framework and Motivation

Despite RBAC being adopted as a standard, it suffers from many shortcomings, including performance. Coyne and Weil bring out many limitations while comparing RBAC with ABAC[13]. Similarly, in their critique of the ANSI standard, charged that though it had been adopted by many vendors, including vendors for database management systems, the RBAC "standard is hindered by limitations, errors, and design flaws"[14].The impacts of an RBAC implementation on performance were recently analysed by C and He who presented their findings in IT Professional journal[15]. A user is often assigned to multiple roles, and in such cases, the security framework ensures that the user's access is controlled by the union of all granted roles[16]. The actual implementation is often achieved through the use of RDSP SQL sub-queries which are pre-created and stored in either in a central grants table in the database or in the new XACML format in the policy store. The security framework ensures that at run time, the required data security sets are put together based on the row-level access grants for a defined role or multiple roles as the case may be. When a user accesses the application dashboards/reports which require SQL queries to be sent to the back-end relational database, all of the user's applicable RDSP sub-queries are compiled together and appended to the main SQL by the security framework. As part of business flows, a user typically defines additional search criteria which are also appended to the main SQL by the application framework. The final SQL thus generated at run time takes the form where the RDSP sub-query could be a simple IN or EXISTS predicate based on a single table reference or a complex set of nested sub-queries based on multiple tables and views.

In modern cloud and SaaS applications, users expect sub-second response times while data security checks add a big performance overhead. This aspect was studied by Kohler and Fies and to overcome some of the performance challenges of RBAC, they suggested a framework that would use pre-computing and caching access decisions in order to meet the performance requirements[17]. Though caching can result in improved performance for the user experience, we think that this solution does not address the fundamental underlying problem.

3. Problem Formulation, Material and Methods

When we benchmarked hundreds of TBI queries for Oracle's Fusion CRM Applications[18], we found that in a majority of the cases the reason for poor performance was the inability of relational database's Cost-Based Optimizer (CBO) to find an optimal execution plan. Our analysis identified the union-all of RDSP sub-queries, appended to data access queries at run time, as the top contributor to the CBO's poor decisions and were the most significant performance inhibitor for the SQL queries. Yaish and Goyal studied the effects of having multiple tenants in a database and each tenant having multiple users. They suggested development of "a multi-tenant access control model based on a multi-tenant database schema" to meet the new challenges[19]. In[20] lamented the slow adoption of Cloud-computing due to security concerns and proposed a comprehensive security framework for cloud computing environments. In[21] proposed a new architecture for fine-grained access control for cloud computing. However, what we found during our study of existing literature of cutting-edge access control technologies from the leading researchers, practitioners and vendors is that there was no existing or proposed architecture that would resolve the performance challenges of row-level data security for enterprise TBI applications.

We found that the problem was compounded by two facts. One, TBI queries, as all other modern business applications' queries, are generated at run-time based on the data model and abstraction layers as well as the design

of dashboards/reports. Invariably, the resulting physical SQL has high complexity with many joins and aggregate functions. This is before RDSP sub-queries are appended based on the user's roles, group memberships and responsibilities. The RDSP sub-queries themselves access multiple tables and views including role and territory hierarchies making them fairly complex by themselves. Thus, with RDSP sub-queries appended to them, the resultant physical SQL queries become very large and complex. The complexity and size of TBI SQL queries along with RDSP sub-queries often cause the optimizer to choose inefficient execution plans and thus negatively impact their performance. We also found that the implementation of RBAC model in the application's security framework led to applying all of a user's applicable RDSPs to all TBI queries, regardless of what dashboard, report or chart was the target of data being fetched by such queries. We concluded that the apparent disconnect between the application and security frameworks was resulting in extremely expensive and wasteful database operations and needed a resolution. This led us to our formulation of the proposed architecture.

4. Discussion and Proposed Architecture

Given the many factors affecting the performance of TBI SQLs that get appended with multi-role RBAC-based RDSP sub-queries, we propose creation of a hand-shake framework that acts as a broker between the TBI application framework and the security framework. The goal of this hand-shake is to trim down the superset of RDSP sub-queries to only the ones deemed necessary by the application framework before the final SQL queries are dispatched to the backend database for execution. The proposed solution encompasses the following:

4.1 At the Security Framework Level

First, for a defined set of Job, Duty and Data roles in an enterprise, create unique identifiers for each role. For example, Opportunity Territory Resource Duty could be identifiedby"FNDDS__MOO_VIEW_OPPORTUNITY_BY_TERRITORY_DATA__MOO_OPTY". Each such identified role corresponds to a RDSP sub-query which can continue to be stored in an existing policy store, either in the relational database or in XACML. Second, create the ability for the security framework to consume name-value pairs when received from the TBI platform/application in order to set the query context and prune the user's applicable RDSPs down to only the ones specified in the name-value pairs received from the TBI platform/application framework. For example, OPTY_DS_CONTEXT='FNDDS__MOO_VIEW_OPPORTUNITY_BY_TERRITORY_DATA__MOO_OPTY'

4.2 At the TBI Platform and Application Framework Level

First, create a special class of standardized security variables to be used for different application entities. For example, Opportunity, Lead etc. in a CRM application could use variables like OPTY_DS_CONTEXT or LEAD_DS_CONTEXT. Second, develop the ability to pass name-value pairs of such security variables to the security framework. For example, a variable named OPTY_DS_CONTEXT could be set with the value 'MOO_VIEW_MY_OPPORTUNITY_DATA' and passed on as XML input to the security framework as in this example: <Parameter><Name><![CDATA[OPTY_DS_CONTEXT]]></Name><Value><![CDATA[FNDDS__MOO_VIEW_OPPORTUNITY_BY_TERRITORY_DATA__MOO_OPTY]]></Value></Parameter>.

4.3 At the User Interface (UI) Layer of the TBI Application

Create user-friendly and easily understood names to be displayed as report prompts. These can be presented to the users in any UI-friendly method like a drop-down pick-list or a multiple selection list or a radio-button choice. The presented list should be restricted and based on the roles available and defined for the user logged in to the application. This can be achieved by caching a user's applicable roles when the user connects to the application. Such caching would prevent any round trips to the policy store as the user navigates to different parts of the application. Also, we propose to provide the means for this name-list to be customized by both the end-user as well as by the application administrator, based on user preferences and/or roles. First, the new class of variables need to be added for which values can be set via dashboard prompts. For example, for the custom value MOO_VIEW_MY_OPPORTUNITY_DATA, we would like to have display name "My Opportunities", such that when

"My Opportunities" is chosen by the user at the dashboard prompt, the variable OPTY_DS_CONTEXT would be set with the value 'MOO_VIEW_MY_OPPORTUNITY_DATA'. The names-values pairs of this new class of variables are then passed on by the middleware server to the security framework, providing the context to selectively apply data security criteria. Similarly, query context can also be set at analysis/report level, instead of dashboard prompts. This variable name-value pair would convey the data security context and can take the form OPTY_DS_CONTEXT = 'FNDDS__MOO_VIEW_OPPORTUNITY_BY_SUBORD_SALES_TEAM_DATA__MOO_OPTY'. When the security framework receives this name-value pair, it prunes down the user's applicable RDSPs to keep only the specified RDSP, thus generating a much simpler, smaller in size physical SQL which almost invariably results in much better performance as compared to the bloated physical SQL in existing architecture. A representative use case from a CRM Application is shown below. In the example, a sales manager navigates to a report titled "View Opportunity by Subordinate Sales Team Data". The leaner, more performant physical SQL generated using our proposed architecture would only keep the one RDSP sub-query based on the name-value pair passed from the TBI application to the security framework. As a result, the final SQL generated using our proposed architecture is much smaller in size and much less complex than the SQL generated in existing architectures. We share the results of one such industry implementation and its benchmark results in the next section.

5. Benchmark Results from Implementation of Proposed Architecture

Our proposed architecture has been recently implemented in Oracle®'s Fusion Transactional Business Intelligence Applications[22] that are built using the Oracle® Application Development Framework (ADF)[23] and Oracle® Business Intelligence Enterprise Edition (OBI EE) OTBI platform[24]. For the benchmarking experiments to establish the effects on performance of our proposed architecture, we used the Lead Management and Opportunity Management modules of Oracle's Fusion CRM OTBI application against an Oracle® 11gR2 database. We used five different users with multiple, varying roles and data access permissions. The benchmark established very promising results. We recorded repeatable, significant gains in not only individual SQL performance but also at the database resources level. Tables 1and 2 show the significant performance gains in SQLs' response time (RT), buffer gets/logical IO, hard parse time and shared memory utilization. Table 1 highlights the SQL Performance Gains for Oracle's CRM OTBI Top Open Deals Report while Table 2 depicts the SQL Performance Gains for Oracle's CRM OTBI Actuals Vs Quota Report.

Table 1. SQL Performance Gains for Oracle's CRM OTBI Top Open Deals Report

CRM OTBI Queries for Top Open Deals Report	Performance Improvement with proposed solution BufferGets	Performance Improvement with proposed solution Cold RT(Includes Hard Parse Time)	Performance Improvement with proposed solution Warm RT
Use Case 1	2X	8X	15%
Use Case 2	43%	6X	6%
Use Case 3	26X	5.6X	6.9X
Use Case 4	5X	39%	5%
Use Case 5	1.8X	21%	12%

Table 2. SQL Performance Gains for Oracle's CRM OTBI Actuals Vs Quota Report

CRM OTBI Queries for Actuals Vs Quota Report	Performance Improvement with proposed solution BufferGets	Performance Improvement with proposed solution Cold RT(Includes Hard Parse Time)	Performance Improvement with proposed solution Warm RT
Use Case 1	2.6X	6X	1.2X
Use Case 2	6X	12.8X	1.5X
Use Case 3	454X	745X	138X
Use Case 4	122X	246X	13X
Use Case 5	17X	2X	2.6X

6. Conclusion

In this paper, we recollected that RBAC based row-level data security has been established as an industry standard practice for many types of applications including enterprise transactional business intelligence applications[25]. While RBAC and XACML implementation details have been very well researched and documented, the impacts of RBAC based row-level data security on the performance and scalability of TBI queries has not been the focus of much

research[26]. Though Ferraiolo, Kuhn and Chandramouli claimed in their book that "Traditionally, commercially available products lag behind the research community. It is always easier to create a new security model or write a paper than to implement and market a new feature." We show in this paper that this need not be the case. We have proposed a novel architecture solution to create a framework that acts as a broker between the security framework and application framework to focus on the trimming down of the superset of RDSP sub-queries to only the ones deemed necessary by the application framework before sending the SQL queries to the backend database.

The benchmark testing of many uses cases in Oracle® Fusion CRM Transactional Business Intelligence Applications against an 11gR2 Oracle® database have shown significant performance gains by adopting the context-sensitive data security proposed in this paper. There are summarized below.

- Query Response Time (RT) improvements ranging from 5% to 745 times for the queries benchmarked, and in some cases where the queries would hang earlier, by many orders of magnitude.
- Database hard parse time improvements from 1% to 208 times.
- Logical I/O or Buffer Gets' improvement ranging from 43% to 454 times.
- SQL-shared-memory reduction by up to 52%.

The proposed improvement over existing RBAC architectures has direct applicability to many SaaS and Cloud applications and can lead to the improvement in their adoption rates. We feel that our work is only a humble beginning and we expect further research to be done on the performance impacts of RBAC solutions in enterprise applications as more and more vendors as well as consumers move towards the Cloud and SaaS model. The aim should be to minimize the impacts of RBAC on the performance of enterprise business applications so that adoption becomes easy.

7. References

1. Ferraiolo DF, Kuhn DR. Role based Access Control' 15th National Computer Security Conf, USA. 1992. p. 554–63.
2. Kuhn DR. Mutual Exclusion of roles as a means of implementing separation of duty in role-based access control systems'. Second ACM Workshop on Role-Based Access Control, Maryland. 1997; 1–8.
3. Sandhu R, Ferraiolo D, Kuhn R. American National Standard for Information Technology – Role based Access Control. ANSI INCITS. 2004; 359:1–49.
4. American National Standards Institute Standards for RBAC. Available from: http://www.incits.org/INCITS_Published_Standards.pdf, Date accessed: 4/03/ 2015.
5. Chandramouli R, Sandhu R. Role based access control features in commercial database management systems'. 21st National Information Systems Security Conference, Crystal City, Virginia. 1998. p. 1–9.
6. Chandramouli R. Business Process Driven Framework for defining an Access Control Service based on Roles and Rules'. 23rd National Information Systems Security Conference, Gaithersburg, 2000. p. 1–16.
7. Ferraiolo DF, Kuhn DR, Chandramouli R. Role Based Access Control (book), Artech House, USA, 2007.
8. Ferraiolo DF, Kuhn DR, Chandramouli R. Role Based Access Control (book), Artech House, 2nd edition, 2007.
9. XACML OASIS Standard, XACML-V3.0, 2013. Available from: http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.pdf, Date accessed: 24/ 03/ 2015.
10. Sun Y, Wang Q, Li N, Bertino E, Atallah MJ. On the Complexity of Authorization in RBAC under Qualification and Security Constraints. IEEE Transactions on Dependable and Secure Computing. 2011; 8(6):883–97.
11. Resource document, 'Oracle® Fusion Applications Security Guide 11g Release 1 (11.1.4)'. Available from: http://docs.oracle.com/cd/E28271_01/fusionapps.1111/e16689/F323388AN16D1F.htm Date accessed: 10/ 07/ 2015.
12. Kohler M, Schaad A. ProActive Access Control for Business Process-Driven Environments. Computer Security Applications Conference, Annual, 2008. p. 153–62.
13. Coyne E, Weil TR. ABAC and RBAC: Scalable, Flexible, and Auditable Access Management, IT Professional. 2013; 15(3):14–6.
14. Li N, Byun J, Bertino E. A Critique of the ANSI Standard on Role-Based Access Control. IEEE Security and Privacy. 2007; 5(6):41–9.
15. Chaudhary N, He L. Analyzing the performance impact of authorization constraints and optimizing the authorization methods for workflows, HIPC. 20th Annual International Conference on High Performance Computing, U K. 2013. p. 1–9.
16. Resource document, 'Oracle® Fusion Applications Security Guide 11g Release 1 (11.1.4). Available from: http://docs.oracle.com/cd/E28271_01/fusionapps.1111/e16689/F323388AN16D1F.htm, Date accessed: 10 / 07/2015.
17. Kohler M, Fies R. ProActive Caching - A Framework for Performance Optimized Access Control Evaluations', IEEE International Workshop on Policies for Distributed Systems and Networks, IEEE International Symposium on Policies for Distributed Systems and Networks, Germany. 2009. p. 92–4.

18. Resource document, Oracle Business Intelligence OTBI Architecture. Available from: http://docs.oracle.com/cd/E51367_01/fa_lcm_gs/OASAD/otbi_trouble.htm#OASAD6512 Date accessed: 12/06/2015.
19. Yaish H, Goyal M. Multi-tenant Database Access Control. IEEE 16th International Conference on Computational Science and Engineering. 2013. p. 870–7.
20. Takabi H, Joshi JBD, Ahn G. Secure Cloud: Towards a Comprehensive Security Framework for Cloud Computing Environments. IEEE 38th International Computer Software and Applications Conference Workshops, USA. 2010. p. 393–8.
21. Msahli M, Chen X, Serhrouchni A. Towards a Fine-Grained Access Control for Cloud, ICEBE. 2014, IEEE 11th International Conference on e-Business Engineering (ICEBE). 2014. p. 286–91.
22. Resource document, Oracle® Fusion Transactional Business Intelligence. Available from: https://docs.oracle.com/cloud/farel8/common/OATBI.pdf, Date accessed: 4/ 06/ 2015.
23. Resource document, Oracle Application Development Framework. Available from: http://www.oracle.com/technetwork/developer-tools/adf/overview/index.html, Date accessed:12/06/2015.
24. Resource document, Oracle Business Intelligence Enterprise Edition product details. Available from: http://www.oracle.com/us/solutions/ent-performance-bi/enterprise-edition-066546.html
25. Gandhi A. Literature Review on Impact of CRM, SRM, Information Sharing and Goal Congruence on Retail-SCM. Indian Journal of Science and Technology. 2016 Jun; 9(22):1–9.
26. Leena NF, Jaykumar V, Issac SS. Assessing CRM Practices in Hotel Industry: A Look at the Progress and Prospects. Indian Journal of Science and Technology. 2015 Mar; 8(S6):1–9.