مجلة جامعة تشرين للبحوث والدراسات العلمية _ سلسلة العلوم الهندسية المجلد (39) العدد (6) 2017
Tishreen University Journal for Research and Scientific Studies - Engineering Sciences Series Vol. (39) No. (6) 2017

Enhancing safe and efficient query processing under the Truman model for fine-grained access control

Dr.Yousef Dalla*
Alaa Akar**

(Received 27 / 9 / 2017. Accepted 20 / 11 / 2017)

□ ABSTRACT □

The increase in sensitive online transactions of organizations across the network has made database an important asset to the organization and thus became a target for internal and external security breaches. Due to the failure of traditional data protection systems, researchers have sought to the Fine-Grained Access Control (FGAC) model to perform this task because of its accuracy in meeting current security requirements at the level of records, columns and even individual cells. Despite the diversity of proposed FGAC methods, they are classified into two main models: Truman & Non-Truman model.

The research is mainly concerned with the study of the problems resulting from the enforcing of the access control policy under the Truman model, we have concentrated on the weakness of the performance of the restricted query and studied the factors affecting it. We have improved the performance by predicate caching , and enhanced the safe processing of queries under this model, reducing the likelihood of leakage through user-defined functions.

Keywords: Access Control, Relational Databases, Database Security, Security Policy function, Truman Model, fine-grained, Query Modification, Data Leakages, Context

*Assistant professor, , Department of Software and Information Systems, Faculty of Informatics Engineering, Tishreen University, Lattakia, Syria.
** Postgraduate student ,Department of Software and Information Systems, Faculty of Informatics Engineering, Tishreen University, Lattakia, Syria.

مجلة جامعة تشرين للبحوث والدراسات العلمية _ سلسلة العلوم الهندسية المجلد (39) العدد (6) 2017
Tishreen University Journal for Research and Scientific Studies - Engineering Sciences Series Vol. (39) No. (6) 2017

تعزيز المعالجة الآمنة و الفعالة للاستعلامات في ظل النموذج Truman للتحكم بالوصول على المستوى الناعم

الدكتور يوسف دلّا*

آلاء اكر**

(تاريخ الإيداع 27 / 9 / 2017. قُبِل للنشر في 20/ 11 / 2017)

□ ملخّص □

إن تزايد التعاملات الإلكترونية للمنظمات الحساسة عبر الشبكة جعل من قواعد البيانات أحد الأصول المهمة للمنظمة ، وأصبحت بالتالي هدفاً للاختراقات الأمنية الداخلية منها والخارجية . و بسبب فشل الأنظمة التقليدية في حماية البيانات توجه الباحثون نحو نموذج التحكم بالوصول على المستوى الناعم Fine-Grained Access Control (FGAC) لأداء تلك المهمة وذلك لما يمتاز به من دقة في تحقيق المتطلبات الأمنية الحالية على مستوى الأسطر و الأعمدة وحتى الخلايا المفردة. وعلى الرغم من تنوع الأساليب المقترحة للنموذج FGAC فإنها تصنف إلى نموذجين رئيسيين هما Truman & Non-Truman model .

قمنا بالاهتمام بشكل أساسي بدراسة المشاكل المترتبة على فرض سياسة التحكم بالوصول في ظل النموذج Truman ، و أهمها ضعف أداء الاستعلام المقيد و درسنا العوامل المؤثرة على ذلك . عملنا على تحسين الأداء عبر تخزين الشروط ، كما عملنا على تعزيز المعالجة الآمنة للاستعلامات في ظل هذا النموذج بما يقلل من احتمالات التسرب عبر التوابع المعرفة من قبل المستخدم .

الكلمات المفتاحية : التحكم بالوصول ، قواعد البيانات العلائقية ، أمن قواعد البيانات ، تابع السياسة الأمنية ، نموذج ترومان ، الحبيبية الناعمة ، تعديل الاستعلام ، تسريب البيانات .

* مدرس- قسم البرمجيات و نظم المعلومات -كلية الهندسة المعلوماتية-جامعة تشرين-اللاذقية-سورية

** طالبة دراسات عليا(ماجستير) . قسم البرمجيات و نظم المعلومات-كلية الهندسة المعلوماتية-جامعة تشرين-اللاذقية-سورية

مقدمة :

سيبقى أمن قواعد البيانات موضوعاً للنقاش طالما استمرت المنظمات بمشاركة بياناتها القيّمة عبر شبكة الانترنت ، وكلما أصبح خبراء الأمن Security Administrators أكثر براعة في مواجهة التهديدات الأمنية كلما أخذت تلك التهديدات أشكالاً أكثر تعقيداً و تطوراً . هذا وإن اعتماد نظم إدارة قواعد البيانات Database Management System(DBMS) في حماية بياناتها الحساسة على طبقات إضافية تبنى عليها أو اكتفائها بتشفير البيانات أثناء إرسالها عبر الشبكة لم يعد مجدياً وينعكس بشكل كبير على الأداء ، الأمر الذي أوصل الخبراء إلى ضرورة حماية البيانات عند مصادرها و داخل قواعدها.

تم اقتراح عدة نماذج لتقييد وصول المستخدمين للبيانات Access Control ، ولكن قصور هذه النماذج عن تنفيذ السياسات الأمنية المعقدة لبعض الشركات من جهة وتعاملها الأمني مع أغراض قاعدة البيانات ككل متكامل من جهة أخرى دفع الباحثين إلى اقتراح وسائل بديلة وشفافة تتحكم بوصول المستخدمين للبيانات على مستويات تفصيلية دقيقة جداً كالأعمدة والأسطر وحتى محتوى الخلية الواحدة ، دعيت تلك الوسائل بالتقنيات الحبيبية الناعمة Fine-Grained Techniques . ويعد الحفاظ على خصوصية المستخدمين privacy أحد الدوافع الرئيسية لهذا النموذج [1,2,3].و هناك العديد من الحالات التي تستلزم هذا النوع من التحكم بالوصول نذكر منها مثلاً:

البنود الأمنية المتعلقة بنظام مصرفي ما ، إذ يمكن للعميل الوصول إلى معلومات رصيده فقط بينما يمكن لأمين الصندوق الوصول لأرصدة جميع الزبائن الذين يتبعون منطقته دون الوصول إلى عناوينهم[1] .

اعتمد المطورون سابقاً على كود التطبيق نفسه في تحقيق التحكم بالوصول من النمط fine-grained ، ولكن ذلك أفضى للعديد من المشاكل من أهمها : تكرار المنطق الأمني للبيانات عندما تكون قاعدة البيانات مشتركة بين عدة تطبيقات وما يترتب على ذلك من صعوبة تحقيق التوافق بين القواعد الأمنية ، و صعوبة الصيانة بسبب توزيع المنطق الأمني عبر أغلب أجزاء التطبيق ، بالإضافة إلى إمكانية تعرض التطبيق للاختراق بعدة وسائل كحقن الاستعلامات وغيرها مما يعرض كامل قاعدة البيانات للخطر ، ناهيك عن أن الأمن على مستوى التطبيق لا يحمي قاعدة البيانات من طرق الولوج الأخرى كالاستعلامات المباشرة و أدوات إصدار التقارير [1,2,4].

ولذلك عمد الباحثون إلى نقل عملية التحكم بالوصول على المستوى الناعم من إطار التطبيقات إلى إطار نظم قواعد البيانات لأن ذلك يضمن مراعاة القواعد الأمنية كاملة بغض النظر عن طبيعة التطبيقات المتفاعلة مع قاعدة البيانات أو طريقة الوصول إليها [2].

أهمية البحث و أهدافه :

يعاني المطوِّر عند تصميم آلية للتحكم بالوصول على المستوى الناعم من صعوبة تفسير السياسات الأمنية الحالية المعقدة والغامضة لبعض المنظمات ، وترجمتها إلى تعليمات برمجية تلتزم بدقة بتمثيلها دون أن تؤثر بشكل حاد على أداء النظام وسرعة استجابته لطلبات المستخدمين ، وتؤمن بنفس الوقت حماية البيانات من عمليات الوصول غير الشرعي . ولسوء الحظ فإننا نفتقد حتى الآن لوجود طريقة للمعالجة الآمنة للاستعلامات دون أن نضطر لفقد بعض المعلومات أو انتهاك السياسة الأمنية أو إعادة نتائج خاطئة للمستخدم .

تتركز أهداف البحث في النقاط التالية :

- دراسة العوامل المؤثرة على أداء الاستعلامات المقيدة في ظل النموذج Truman والتركيز على العامل المتعلق بتعقيد السياسة الأمنية للتحكم بالوصول .
- تحسين فعالية الاستعلامات المقيدة وتقييمها وفق عدة حالات.
- تعزيز المعالجة الآمنة للاستعلامات في ظل النموذج Truman للتحكم بالوصول على المستوى الناعم FGAC ، بما يقلل من احتمالات التسرب وخصوصاً عبر التوابع المعرفة من قبل المستخدم .

طرائق البحث و مواده :

تم اتباع المنهج التجريبي و الاستقصائي في البحث ، حيث اعتمدنا على Oracle DBMS كأداة برمجية لإنجاز البحث. تم تصميم عدة سيناريوهات لاختبار أداء الاستعلامات المقيدة بسياسة النموذج truman للتحكم بالوصول و ذلك لكشف مشاكله و اقتراح حلول لها . و اعتمدنا في تجاربنا على بيانات متفاوتة الحجوم و على عدة متغيرات متعلقة بموضوع البحث تم اقتباس بعضها من الدراسات المرجعية السابقة كما تم اقتراح بعضها الآخر . ومن أبرز العوامل المدروسة تعقيد السياسة الأمنية المستخدمة للتحكم بالوصول pf complexity و هو الزمن اللازم لتنفيذ البنى البرمجية المسؤولة عن تطبيق السياسة الأمنية للتحكم بالوصول . أما المقياس الذي اعتمدناه لأداء الاستعلامات فهو سرعة الاستجابة لطلبات المستخدمين و الناتج عن الزمن الإجمالي اللازم لتنفيذ الاستعلام وإعادة النتائج للمستخدم مقدراً بالميلي ثانية (ms) وسنصطلح عليه بزمن التنفيذ.

1- نموذج ترومان للتحكم بالوصول :

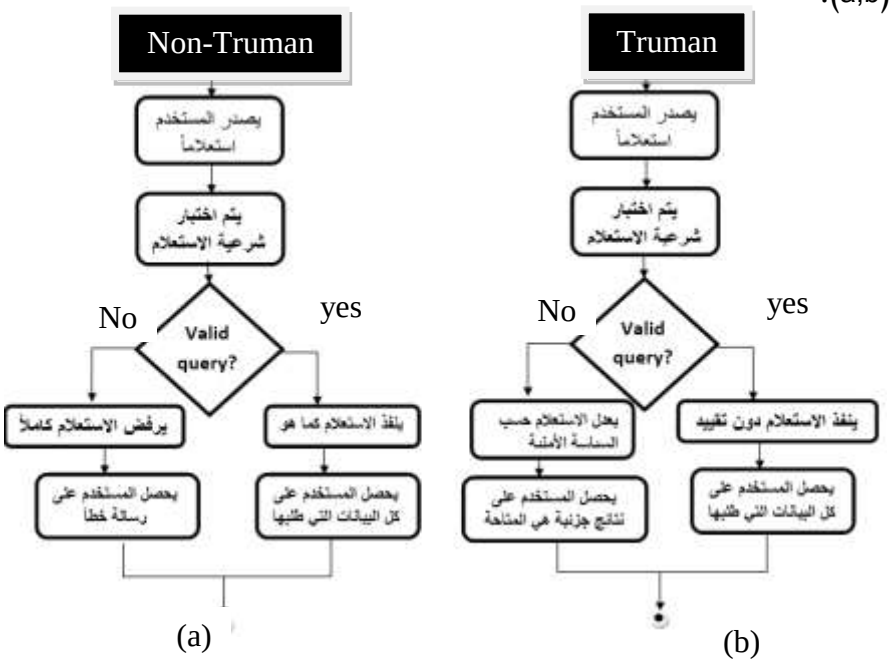
على الرغم من تنوع الأساليب المقترحة لتحقيق النموذج FGAC فإنها تندرج تحت تصنيفين رئيسين تم وضعهما من قبل Rizvi [1] وهما Truman Model & Non-Truman Model .

يقوم نموذج Truman على فكرة تزويد كل مستخدم بمنظار شخصي محدد لكل علاقة في قاعدة البيانات . يعرّف المنظار كل ما يمكن للمستخدم الوصول إليه في تلك العلاقة ولذلك يدعى بمنظار التخويل Authorization View يقوم نظام DBMS هنا بالتحكم بوصول المستخدمين إلى البيانات المتاحة لهم بشكل شفاف . إذ يمكن للمستخدم توجيه استعلاماته إلى العلاقات الأساسية في قاعدة البيانات و يتولى نظام DBMS عندها مسألة تعديل الاستعلام الأصلي بشكل آلي من خلال استبدال كل علاقة واردة فيه بالمنظار الموافق لهذا المستخدم و المنسجم مع السياسة الأمنية للمنظمة .[1] وعلى الرغم من أهمية النموذج Truman فإن عملية تعديل الاستعلام التي يقوم بها تعاني من مجموعة من المشكلات أهمها إمكانية حصول المستخدم على نتائج مضللة ومتناقضة مع توقعاته بالإضافة إلى زيادة تعقيد الاستعلام المعدل وإمكانية وجود شروط فائضة قد يسبب اختبارها تأخيراً كبيراً وغير مبرر في زمن تنفيذ الاستعلام المقيد. [5] إن هذه الأسباب و غيرها دفعت الباحثين للتفكير بنموذج آخر يعمل على تجاوز المشكلات المترتبة على تعديل الاستعلام .

1-1- نموذج Non-Truman للتحكم بالوصول :

تخضع الاستعلامات في إطار هذا النموذج إلى اختبار صلاحية validity test تحدد نتيجته أسلوب تنفيذها ، فعندما يتجاوز الاستعلام هذا الاختبار يتم تنفيذه كما هو دون أي تعديل ، ولكن عندما لا يتم الاقرار بصلاحية الاستعلام يتم رفضه بالكامل وإخبار المستخدم بذلك من خلال التقاط استثناء برمجياً أو رسالة خطأ ما . يتم الإقرار

بصلاحية الاستعلام عندما يستهدف بيانات متاحة له وفق السياسة الأمنية و مضمّنة في المناظير المخصّصة للمستخدم المعني [1]. يمكن بالاعتماد على ماسبق تلخيص الفرق بين النموذجين Truman و Non–Truman من خلال الشكل (1) (a,b):



الشكل(1) آلية عمل النموذجين Truman & Non–Truman

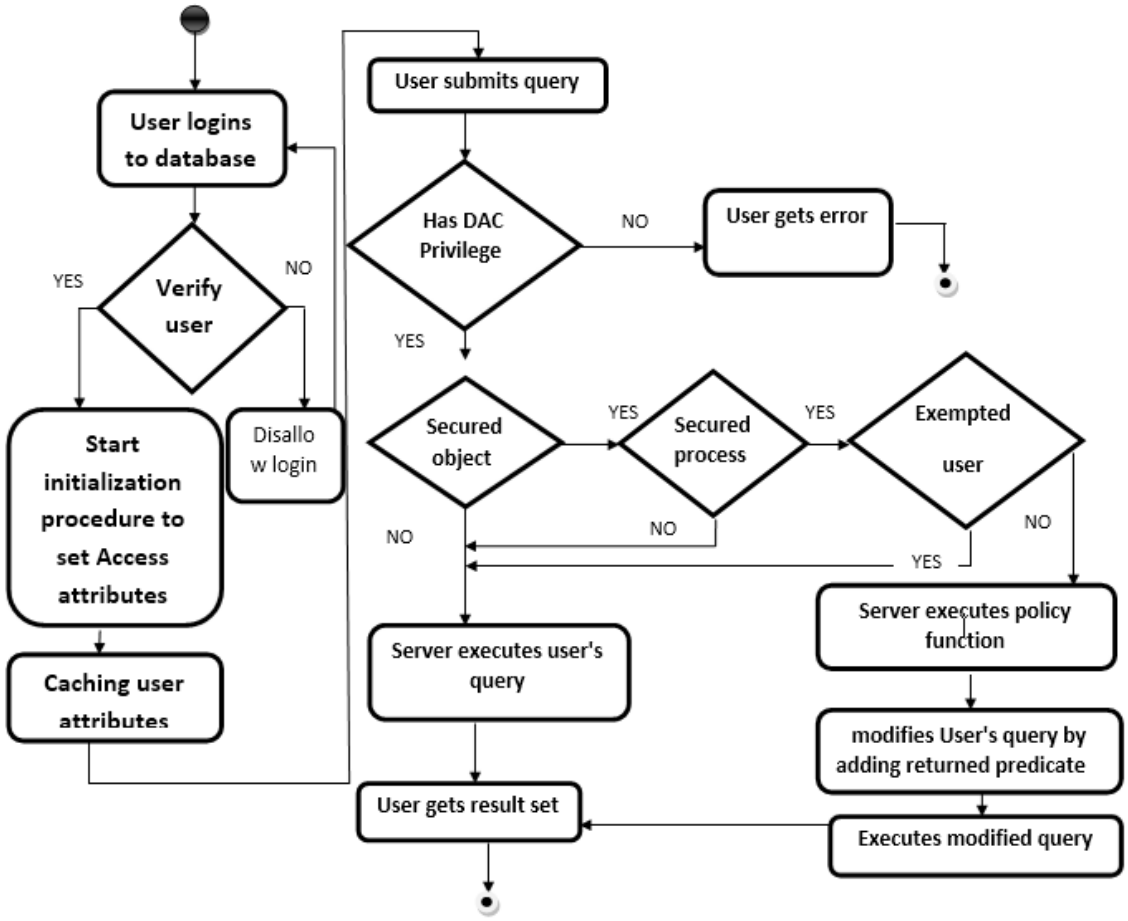
1–2– الخوارزمية المتبعة لتحقيق النموذج Truman في البحث :

يمكن تنفيذ النموذج Truman برمجياً بإحدى طريقتين : الأولى باستخدام مناظير التخويل Authorization Views و الثانية باستخدام أسلوب تعديل الاستعلام بإضافة الشروط المقيدة للوصول ، وهي الطريقة المتبعة في البحث. وقد اعتمدنا في تحقيقها على تقنية VPD [1]التي تعمل على تقييد عمليات الوصول الى البيانات وفق سياسة أمنية معينة security policy. ويقصد بالسياسة الأمنية هنا مجموعة القواعد التي تحكم عملية الوصول إلى البيانات وفق رؤية الجهة المالكة للبيانات . يقوم المدير الأمني لقاعدة البيانات بالتعبير عن قواعد الوصول من خلال تابع PL/SQL[2] يدعى تابع السياسة الأمنية security policy function يُربط مع الغرض الذي نرغب بحمايته . وعندما يقوم المستخدم بإصدار استعلام ما على الغرض المحمي تلحق السلسلة النصية المعادة من هذا التابع (والتي تكون بمثابة شرط predicate) بعبارة SQL الأساسية التي أصدرها المستخدم لنحصل على استعلام معدّل جديد يقيد وصوله إلى البيانات المتاحة له فقط وفق السياسة الأمنية المفروضة. يمكن فرز المعلومات التي يمكن للمستخدم الوصول إليها بناءً على عدة عوامل تتعلق بهوية المستخدم مصدر الاستعلام أو محتوى البيانات نفسها أو حتى عوامل أخرى تتعلق ببيئة العمل. تمتاز تقنية VPD بخطوات عامة يقوم المطور ببرمجتها و صياغة الشروط الملائمة لمتطلبات التطبيق الأمنية [7]. يبين الشكل (2) مخطط النشاط Activity Diagram الخاص بالتحقيق البرمجي الذي

[1] **VPD : virtual private database تقنية للتحكم بالوصول من الحزمة الأمنية الخاصة بنظام oracle لإدارة قواعد البيانات .**

[2] **Pl/SQL : Procedural Language/Structured Query Language اللغة الإجرائية المدمجة بلغة الاستعلامات البنيوية.**

اتبعناه للنموذج Truman في نظام Oracle DBMS ويبين بوضوح خطوات الخوارزمية التي ستتولى معالجة الاستعلام المقيد بسياسة تحكم بالوصول.



الشكل(2): مخطط النشاط الخاص بالتحقيق البرمجي للنموذج Truman

1-3- الدراسة التجريبية للنموذج Truman للتحكم بالوصول على المستوى الناعم :

إن أي نموذج مقترح لتحقيق التحكم بالوصول على المستوى الناعم FGAC يجب أن يأخذ بالحسبان مسألتين رئيسيتين هما : أداء الاستعلام المقيد و الحصانة ضد تسريب البيانات[5].

1-3-1- أداء الاستعلام المقيد بسياسة تحكم بالوصول على المستوى الناعم:

تهتم أغلب النماذج المقترحة للتحكم بالوصول على المستوى الناعم FGAC بفكرة خلو نتيجة استعلام المستخدم من بيانات تنتهك السياسة الأمنية ، ولكن هل يمكن الاكتفاء بنتيجة الاستعلام المعدل كهاجسٍ رئيسي عند فرض سياسة التحكم بالوصول؟ ماذا عن تأثير عملية التعديل نفسها على أداء الاستعلام؟ إن الهدف من الآلية المستخدمة لتعديل الاستعلام لا يقتصر على إيجاد صيغة الاستعلام الجديد فحسب وإنما أن يكون الاستعلام الجديد فعالاً من ناحية التنفيذ بمعنى ألا يزيد تعقيده [1] .

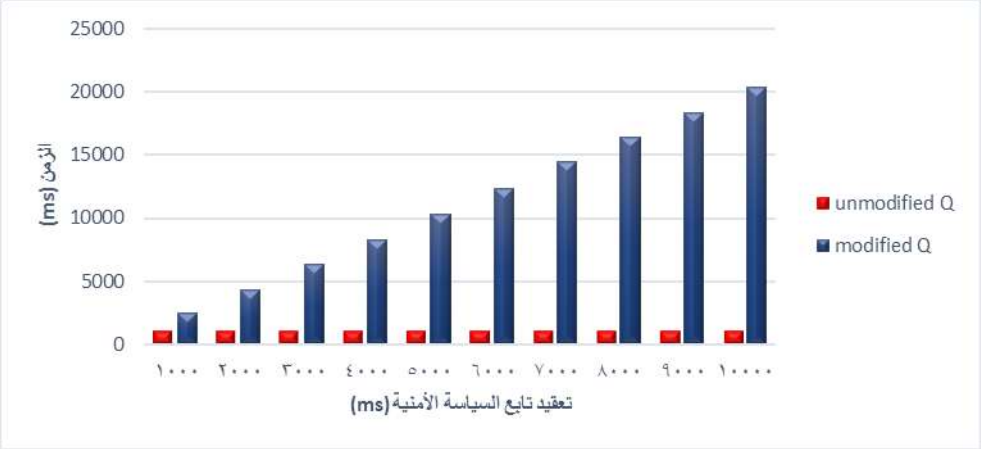
هناك العديد من العوامل التي تؤثر على أداء الاستعلام المعدل(سرعة الاستجابة) كحجم الجدول و عدد الواصفات الحساسة و نسبة الأسطر التي يستهدفها الاستعلام بالإضافة إلى تعقيد تابع السياسة الأمنية. ولكون العوامل الأولى ذات تأثير مشترك على الاستعلامات قبل و بعد تطبيق السياسة ولا يمكن التحكم بها لكونها متعلقة بطبيعة التطبيق بحد ذاته ، فإننا سنركز هنا على دراسة تأثير تعقيد تابع السياسة الأمنية على أداء الاستعلام المعدّل.

1-3-1-1- **الاختبار الأول :** السيناريو المستخدم موضح بالجدول(1) :

الجدول(1): سيناريو دراسة تأثير تعقيد تابع السياسة الأمنية على أداء الاستعلامات المعدّلة

هدف الإختبار	دراسة تأثير تعقيد تابع السياسة الأمنية على أداء الاستعلامات المعدّلة
متغيرات الاختبار	حجم الجدول : 10000 سجل. عدد الواصفات الحساسة[1] : 1 100 : [2]selectivity pf_complexity : متزايد تريجياً في المجال ms [1000..10000]
الحالات المختبرة	Modified Q : الاستعلام المعدل بموجب السياسة الأمنية. Unmodified Q : الاستعلام الأصلي قبل تطبيق السياسة الأمنية.
ظروف الاختبار	يتم تفريغ ذاكرة الاستعلامات المؤقتة shared-pool[3] من الاستعلامات المعربة بعد كل زيادة في تعقيد تابع السياسة الأمنية. مع تنفيذ الاستعلام نفسه ثلاث مرات على جداول متساوية الحجم و أخذ الزمن الوسطي للأزمنة الثلاث.

ونتيجة الاختبار الأول موضحة بالشكل (3).



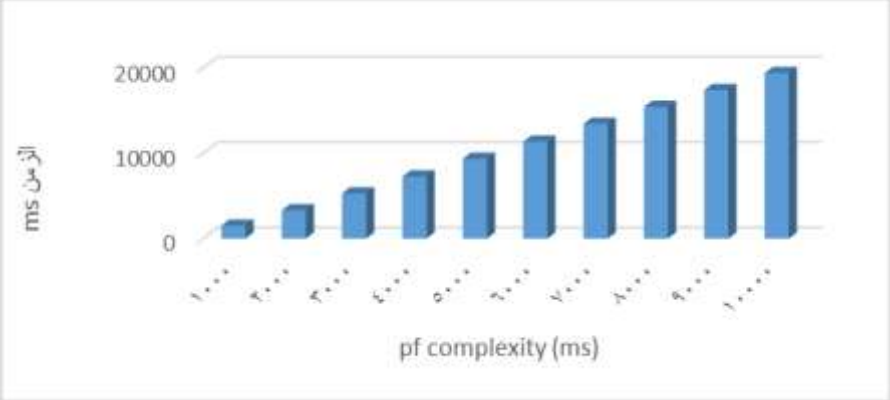
الشكل(3): تأثير تعقيد السياسة الأمنية على زمن تنفيذ الاستعلام المقيّد

يتضح من الشكل(3) أن زمن تنفيذ الاستعلامات المقيدة ينمو بسرعة كبيرة جداً مع زيادة تعقيد تابع السياسة الأمنية مما يعني أن لتعقيد تابع السياسة الأمنية تأثيراً سلبياً كبيراً على سرعة الاستجابة . لو أخذنا الفرق بين الزمن اللازم لتنفيذ الاستعلام الأصلي و الزمن اللازم لتنفيذ الاستعلام المقيد لحصلنا على الشكل(4) الذي يمثل العبء المترتب على فرض السياسة الأمنية (overhead).

[1] **ويقصد بها عدد الواصفات السرية في علاقة ما والتي يتم ضبط الوصول إلى بياناتها بموجب سياسة للتحكم بالوصول .**

[2] **وهي نسبة الأسطر المستهدفة باستعلام المستخدم إلى مجموع الأسطر الموجودة في العلاقة المحمية بسياسة أمنية للتحكم بالوصول .**

[3] **وهي منطقة من الذاكرة مخصصة لتخزين الاستعلامات السابقة مع خطط تنفيذها.**



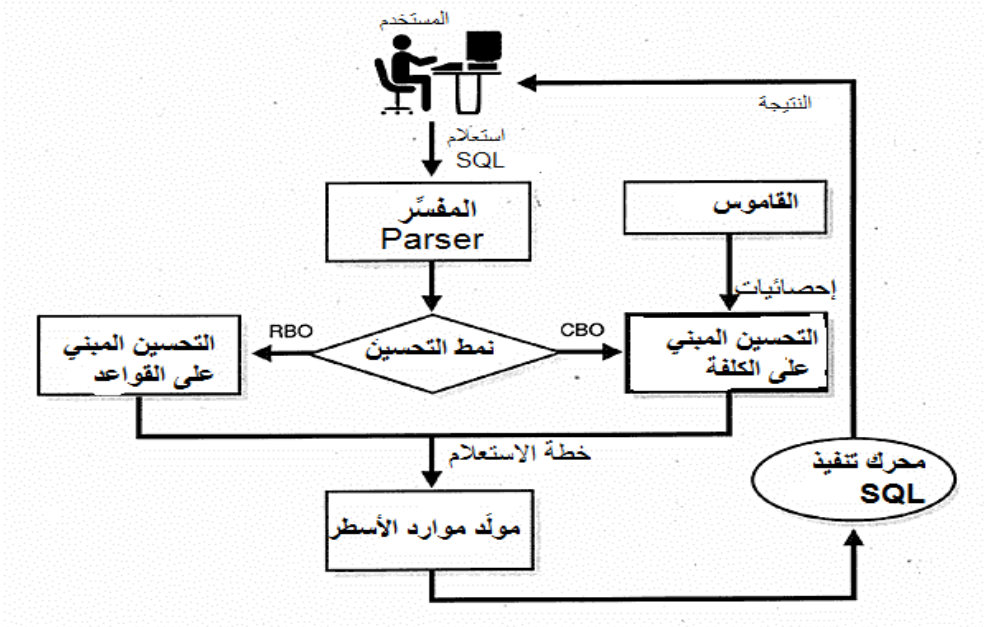
الشكل(4): العبء المترتب على فرض سياسة التحكم بالوصول

يتضح من الشكل(4) أن العبء المترتب على فرض سياسة التحكم بالوصول من النمط FGAC في ظل النموذج Truman يتناسب طرداً مع تعقيد التابع المعبر عن بنودها(policy function).

قمنا بداية بدراسة مراحل معالجة الاستعلام لمعرفة أسباب العبء المترتب على فرض سياسة التحكم بالوصول .

1-3-2- مراحل معالجة الاستعلام :

إن أي عبارة sql يصدرها المستخدم تمر بمراحل المعالجة الموضحة بالشكل (5):



الشكل(5): مراحل معالجة الاستعلام [6]

و فيما يلي شرح مبسط و موجز لما تقوم به كل مرحلة :

1- **مرحلة الإعراب parsing :** و فيها يتم تقسيم عبارة sql إلى مكوناتها الأساسية و القيام بالتحليل القواعدي وتحليل المعنى لها للتأكد من توافقيتها مع معيار sql الأساسي و من أغراض البيانات الواردة فيها . ينتقل الـ parser بعد ذلك إلى البحث في منطقة خاصة من الذاكرة تدعى shared_pool مخصصة لتخزين الاستعلامات بشكل مؤقت . حيث يقوم بالبحث عن وجود عبارات مطابقة للعبارة الحالية جرت معالجتها سابقاً ، فإذا حصل و وجدت مثل تلك العبارات فهذا يعني تجاوز المرحلتين التاليتين للإعراب وهما مرحلتي التحسين optimizing و توليد موارد

الأسطر row source generation ، وبالتالي تحسناً كبيراً في سرعة الاستجابة ، و نقول هنا أننا في حالة الاعراب الناعم soft parse . أما إذا لم نجد مثل تلك العبارات فإننا سنضطر للتحسين و توليد الخطة الأمثلية و بالتالي نقول أننا في حالة الاعراب الشاق hard parse . وبسبب الكلفة العالية المترتبة على عمليتي التحسين و توليد الخطة الأمثلية فإننا نسعى لجعل النسبة الأكبر من استعلاماتنا تسلك النوع soft parse .

2- **مرحلة التحسين Optimizing** : تهدف هذه المرحلة إلى إيجاد الخطة المثالية لتنفيذ عبارة sql من خلال تجريب أكثر من مسار للتنفيذ وحساب تكلفته ، واختيار المسار الذي يعيد النتائج للمستخدم بأقصر زمن و أقل كلفة كخطة مثالية. وتعد هذه المرحلة الأكثر كلفةً بالنسبة لجميع مراحل معالجة الاستعلام.

3- **مرحلة توليد موارد الأسطر Row source generating** : تعمل هذه المرحلة على تحويل الخطة المثالية الى خطة تنفيذ execution plan مؤلفة من مجموعة من الخطوات المتتالية، خرج كل خطوة عبارة عن row source إما ان يكون دخل للخطوة التالية في الخطة أو أن يكون جزءاً من النتائج التي تعاد للمستخدم في الخطوة الأخيرة .

4- **محرك تنفيذ Sql execution Engine (sql)** : الذي يتلقى عبارة sql مرفقة بخطة تنفيذها ليعمل على تلك الخطة في جلب الأسطر من أماكن تخزينها الفيزيائية وإصدار نتيجة الاستعلام للمستخدم.[6]

1-3-3- تحسين أداء الاستعلامات المقيدة بسياسة أمنية للتحكم بالوصول:

إن تنفيذ تابع السياسة الأمنية يستهلك مقداراً كبيراً من موارد النظام . و لتحسين أداء الاستعلامات المقيدة بسياسة أمنية يجب العمل على تقليل عدد مرات تنفيذ تابع السياسة المقابل . و إذا نظرنا إلى الاستعلام المقيد بسياسة أمنية (الاستعلام المعدّل) على أنه مكون من الاستعلام الأصلي مضافاً له الشرط predicate الناتج عن سياسة التحكم بالوصول ، فإنه لتحسين الأداء يجب التأكد من فعالية هذا الشرط . وفي سعينا لتحسين أداء الشرط اعتمدنا على استخدام متحولات الربط bind variables لكونها مؤشراً مهماً وميزاناً للأداء ، لأنها تسمح بمشاركة خطة تنفيذ واحدة من أجل العديد من عبارات الـ sql ، وبالتالي تجعل العبارة الحالية في حالة الإعراب الناعم soft parse.[8] إن استخدام سياق التطبيق application context. كجزء من الشرط المعاد من خلال تابع السياسة الأمنية تجعله يتصرف تماما كَـ bind variable من ناحية استخدام خطة التنفيذ نفسها من أجل عدد كبير من عبارات الـ sql المولدة ، مما يسمح بتحسن الأداء بشكل كبير جدا عن الحالة التي تستخدم فيها القيمة الحرفية للأعمدة الواردة في شرط السياسة الأمنية .

1-3-3-1- استخدام Application Context لتحسين أداء سياسات التحكم بالوصول:

يمكن اعتبار سياق التطبيق بمثابة فضاء عنونة يمكن استخدامه مع سياسات التحكم بالوصول لتحسين أدائها. فهو يستخدم في تعريف القواعد المستخدمة في فصل المستخدمين لعدة مستويات تتفاوت في حقوقها بالوصول إلى البيانات ، حيث يقوم بتخزين آمن لخصائص معينة عن البيانات أو المستخدمين تستخدم في اتخاذ قرار الوصول إلى البيانات ، يمكن أن تتعلق هذ الخصائص بهوية المستخدم أو محتوى البيانات أو نوعية العمليات التي ينفذها المستخدمون على البيانات وغير ذلك. و سنصطلح على تسمية هذه الخصائص بواصفات السياق Context Attributes ويتم تخزينها طيلة الجلسة و تحديث قيمتها مع بداية كل جلسة جديدة . يمكن تهيئة واصفات الوصول هذه من خلال حزمة مخصصة لهذا الغرض سنصطلح على تسميتها بمدير السياق context manager مما يضمن

عدم تغييرها بطريقة غير شرعية. وحيث أن سياق التطبيق يعمل على تخبئة caching تلك الواصفات فإن الأداء سيتحسن بالتأكيد كما سنبين لاحقاً.

كما يسمح استخدام سياق التطبيق في صياغة شرط عام مناسب لكل المستخدمين بدلاً من صياغة شرط خاص بكل مستخدم على حدى .

سنصطلح فيما يلي على تسمية السياسات التي نضطر فيها لتنفيذ تابع السياسة الأمنية عند كل استعلام على الغرض المحمي لتوليد شرط جديد بالسياسات الديناميكية ، بينما السياسات التي تستخدم سياق التطبيق كجزء من الشرط العام المولد من تابع السياسة الأمنية بالسياسات الستاتيكية.

وبهدف قياس التحسن في الأداء الناجم عن استخدام سياق التطبيق قمنا بسلسلة اختبارات قسنا فيها الزمن اللازم لتنفيذ الاستعلامات المقيدة مقدراً بالميلي ثانية (ms) نورد نتائجها فيما يلي :

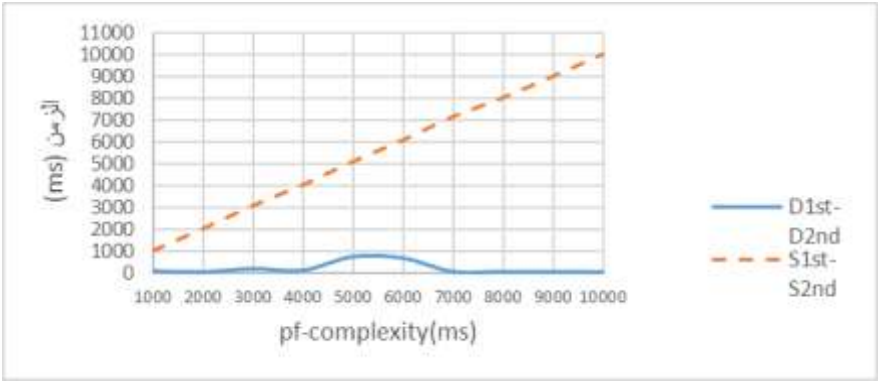
1-3-3-1-1- الاختبار الثاني : السيناريو المتبع في الاختبار موضح من خلال الجدول(2)

الجدول(2): السيناريو المستخدم لإيجاد التسريع الناتج عن استخدام السياسات الستاتيكية

هدف الإختبار	ايجاد التسريع الناتج عن استخدام السياسات الستاتيكية بدلا من السياسات الديناميكية
متغيرات الاختبار	حجم الجدول المستخدم : 30000 record تعقيد تابع السياسة الأمنية (pf_complexity) متزايد تدريجياً بين 1000-10000 ms
الحالات المختبرة	Dynamic_1st : استعلام مقيد بسياسة ديناميكية عند أول تنفيذ له. Static_1st : استعلام مقيد بسياسة ستاتيكية عند أول تنفيذ له. Dynamic_2nd : استعلام مقيد بسياسة ديناميكية عند ثاني تنفيذ له وبنفس الجلسة. Static_2nd : استعلام مقيد بسياسة ستاتيكية عند ثاني تنفيذ له و بنفس الجلسة. Static_new_session : استعلام مقيد بسياسة ستاتيكية عند تنفيذه بجلسة عمل جديدة ومن قبل مستخدم جديد (هيكلية جديدة schema).
ظروف الاختبار	- يتم بداية اختبار السياسات الديناميكية ثم السياسات الستاتيكية . - يتم تنفيذ نفس الاستعلام مرتين متتاليتين في الجلسة الواحدة بدون تفريغ ذاكرة الاستعلامات المؤقتة بين التنفيذين المتتاليين بغرض قياس العبء المترتب على توليد الشرط . - يتم تفريغ ذاكرة الاستعلامات المؤقتة عند كل زيادة في تعقيد تابع السياسة الأمنية policy function و البدء بجلسة عمل جديدة بهدف تهيئة واصفات السياق من جديد.

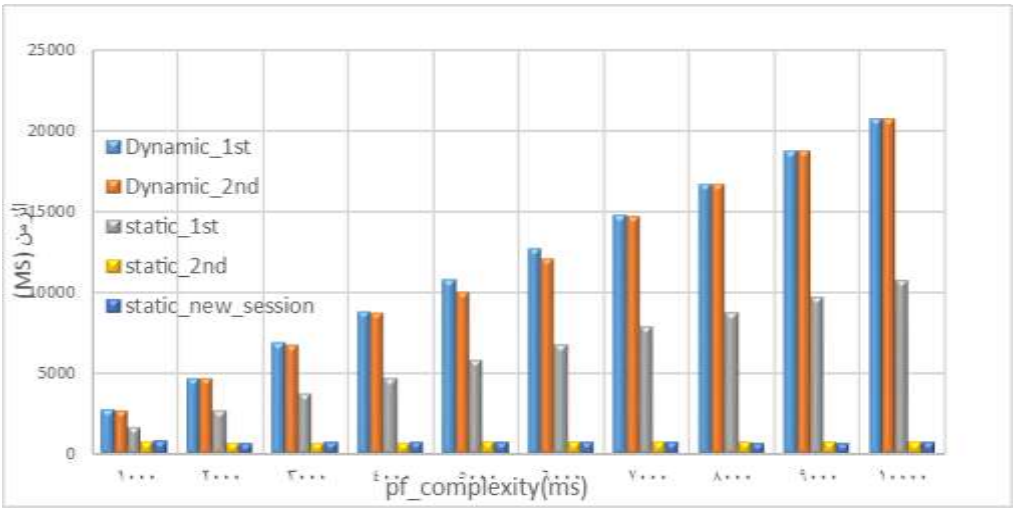
ولمناقشة نتائج الاختبار الثاني قمنا بداية بقياس التسريع الحاصل في التنفيذ الثاني للاستعلام بنفس الجلسة حيث رمزنا لهذا التسريع بـ : (D1st-D2nd) في حالة الشروط الديناميكية و(S1st-S2nd) في حالة الشروط الستاتيكية ، و بالنظر للشكل (6) الناتج نجد أنه عند تكرار تنفيذ الاستعلام نفسه في نفس الجلسة فإن الزمن سيقل بشكل لا يذكر في حالة الشروط الديناميكية ، إذ سنجد تحسناً في الأداء لا يتجاوز الثانية في أفضل حالاته مع جدول متوسط الحجم يضم 30000 سجل ، وهذا لا يقارن مع تحسن الأداء في حال استخدام الشروط الستاتيكية (تخزين

الشرط) ، و يزداد هذا التحسن بازدياد تعقيد تابع السياسة الأمنية بسبب الاستغناء عن تكرار تنفيذه مع كل استعلام جديد و الاكتفاء بتنفيذه لمرة واحدة فقط.



الشكل(6): الكسب الحاصل في التنفيذ المتتالي للاستعلامات عند تخزين الشرط

ويمكن إجمال النتائج التي حصلنا عليها في الاختبار الثاني من خلال الشكل(7) :



الشكل(7) : تحسين أداء الاستعلامات عند تخزين الشرط

بالعودة إلى الشكل(7) نجد أن الزمن اللازم لتنفيذ الاستعلام يزداد مع زيادة تعقيد تابع السياسة الأمنية بغض النظر عن تخزين الشرط أو توليده دائماً بشكل ديناميكي ، و لكن نلاحظ أن الزمن اللازم لتنفيذ الاستعلام في حالة التوليد الديناميكي للشرط ينمو بشكل أكبر بكثير مع تزايد تعقيد تابع السياسة الأمنية عما هو عليه الحال مع تخزين الشرط و إعادة استخدامه ، ويعتبر هذا نتيجة طبيعية فتابع السياسة الأمنية في حال استخدام الشروط الديناميكية يتم تنفيذه مرتين الأولى أثناء الإعراب و الثانية أثناء التنفيذ ، و بالتالي فإن زيادة تعقيد تابع السياسة الأمنية الديناميكية يؤدي إلى زيادة مضاعفة في زمن تنفيذ الاستعلام على الجدول الذي تحميه بسبب العبء الناتج عن تنفيذ تابع السياسة الأمنية مع كل استعلام جديد على الجدول المحمي و الذي سنوفره في حالة تخزين الشرط .

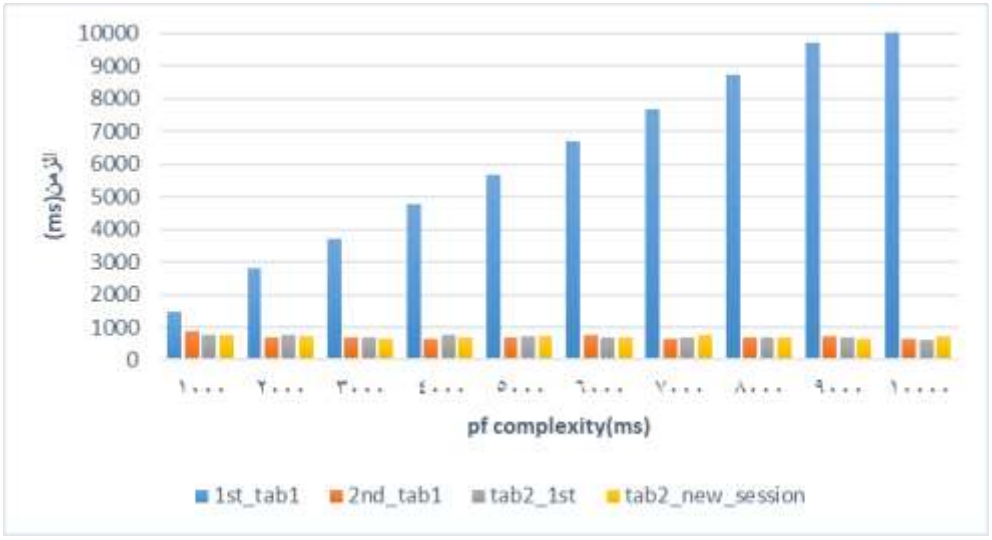
1-3-3-1-2- الاختبار الثالث : السيناريو المتبع في الاختبار موضح من خلال الجدول(3)

الجدول(3): السيناريو المستخدم لقياس التحسن في الأداء للسياسات الساكنة المطبقة على أكثر من جدول

هدف الإختبار	تقييم أداء الاستعلامات المقيدة في ظل السياسة الساكنة المطبقة على أكثر من جدول ضمن أكثر من schema
متغيرات الاختبار	حجم الجدول : 30000 سجل.

	عدد الجداول المستخدمة : 2 بنفس الحجم و ضمن هيكليتين مختلفتين و خاضعين لنفس السياسة. 100 : selectivity
الحالات المختبرة	1st_tab1 : أول تنفيذ للاستعلام في الجلسة على الجدول الأول . 2nd_tab1 :ثاني تنفيذ للاستعلام في الجلسة نفسها على الجدول الأول . tab2_1st : أول تنفيذ للاستعلام في الجلسة نفسها على الجدول الثاني. tab2_new_session : ثاني تنفيذ للاستعلام على الجدول الثاني و بجلسة عمل جديدة.
ظروف الاختبار	- يتم بداية اختبار السياسات الديناميكية ثم السياسات الستاتيكية . - يتم تنفيذ نفس الاستعلام مرتين متتاليتين في الجلسة الواحدة بدون تفريغ ذاكرة الاستعلامات المؤقتة Shared-pool بين التنفيذين المتتاليين بغرض قياس العبء المترتب على توليد الشرط . - يتم تفريغ ذاكرة الاستعلامات المؤقتة Shared-pool عند كل زيادة في تعقيد تابع السياسة الأمنية policy function و البدء بجلسة عمل جديدة بهدف تهيئة واصفات السياق من جديد.

النتائج التجريبية للاختبار الثالث موضحة بالشكل(8) و نستنتج منه أن الزمن اللازم لتنفيذ الاستعلامات ينمو بمقدار متناسب مع زيادة تعقيد تابع السياسة الأمنية ، و أن هذه الزيادة مقصورة على المرة الأولى التي ينفذ فيها الاستعلام على الجدول المحمي ، في حين أن تعقيد تابع السياسة الأمنية ليس له أي تأثير على التنفيذات المتعاقبة للاستعلام نفسه ضمن نفس الجلسة أو بجلسة عمل جديدة أو حتى على جدول آخر محمي بالسياسة نفسها وموجود في هيكلية أخرى ، حيث أننا نجد من المخطط الأول أن الزمن اللازم لتنفيذ مثل هذه الاستعلامات يبقى ثابتاً تقريباً ولا يصل لأكثر من ثانية في ذروته مع حجم جدول متوسط مؤلف من 30000 سجل . و ذلك بسبب تنفيذ تابع السياسة الأمنية مرة واحدة فقط أثناء تنفيذ أول استعلام على أي من الجداول المحمية بموجبه ليتم بعد ذلك تخزين الشرط المعاد من التابع لإعادة استخدامه مع الاستعلامات التالية على أي من الجداول المحمية بالتابع نفسه وفي أي schema كانت .



الشكل(8) : تحسين الأداء باستخدام السياسات الستاتيكية المطبقة على عدة جداول

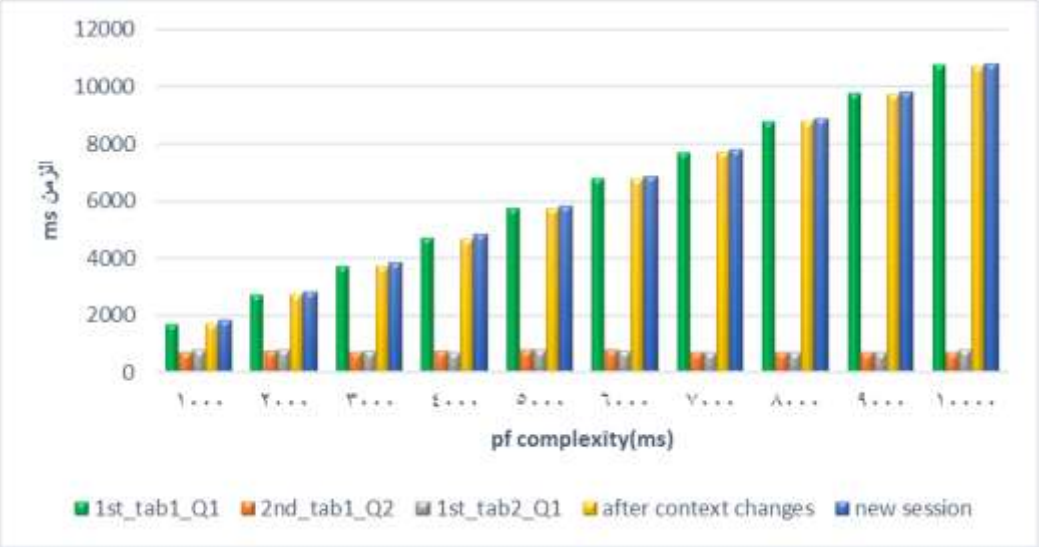
1-3-3-1-3- الاختبار الرابع :معالجة التغيرات في السياق عند تطبيق السياسات الساكنة على عدة جداول:

السؤال الذي نود الإجابة عنه هنا : ماذا سيحدث للشرط المخزن عندما يحصل تغيير مفاجئ في سياق التطبيق مما قد يؤثر بشكل مباشر على طبيعة الشرط؟ ويوضح الجدول(4) السيناريو المتبع في هذا الاختبار.

الجدول(4): معالجة التغيرات المحتملة في السياق عند تخزين الشرط

هدف الإختبار	دراسة الاستعلامات المقيدة باستخدام سياسة ساكنة مطبقة على أكثر من جدول ضمن أكثر من schema ولكن مع مراعاة التغيير المحتمل في سياق التطبيق أثناء الجلسة .
متغيرات الاختبار	حجم الجدول : 30000 سجل. عدد الجداول المستخدمة : 2 بنفس الحجم و ضمن هيكليتين مختلفتين و خاضعين لنفس السياسة. selectivity : 100
الحالات المختبرة	1st_tab1_Q1 : أول تنفيذ للاستعلام في الجلسة على الجدول الأول . 2nd_tab1_Q2 :ثاني تنفيذ لاستعلام آخر في الجلسة نفسها على الجدول الأول . 1st_tab2_Q1: أول تنفيذ للاستعلام في الجلسة نفسها على الجدول الثاني. after_context_changes : تنفيذ الاستعلام على أي من الجداول بعد حدوث تغيير في قيم واصفات الوصول. new_session : ثاني تنفيذ للاستعلام على الجدول الثاني و بجلسة عمل جديدة.
ظروف الاختبار	- يتم بداية اختبار السياسات الديناميكية ثم السياسات الستاتيكية . - يتم تنفيذ نفس الاستعلام مرتين متتاليتين في الجلسة الواحدة بدون تفريغ ذاكرة الاستعلامات المؤقتة Shared-pool بين التنفيذين المتتاليين بغرض قياس العبء المترتب على توليد الشرط يتم تفريغ ذاكرة الاستعلامات المؤقتة Shared-pool عند كل زيادة في تعقيد تابع السياسة الأمنية policy function و البدء بجلسة عمل جديدة بهدف تهيئة واصفات السياق من جديد.

نتائج الاختبار الرابع : موضحة بالشكل (9):



الشكل(9): أداء الاستعلامات المقيدة في ظل السياسات الستاتيكية عند تغيير السياق

يتضح من الشكل (9) أن عملية تخزين الشرط في السياسات الساكنة يبطل مفعولها عند حصول أي تغيير في قيم واصفات السياق ، أو عند البدء بجلسة عمل جديدة حيث يعاد عندها تنفيذ تابع السياسة الأمنية للحصول على الشرط مجدداً حتى ولو لم تكن تلك التغييرات في واصفات السياق مرتبطة بشكل مباشر مع تابع السياسة الأمنية ، و يدل هذا بالتالي على أن تخزين الشرط في هذه الحالة سيكون على مستوى الهيكليات فقط وليس على مستوى الجلسات.

1-3-2- مسألة تسريب البيانات : على الرغم من إمكانية فرض تحكم بالوصول بالغ الدقة عبر النموذج Truman ، فإنه ما زال أمام المستخدم فرصة ليصل إلى بيانات حساسة غير متاحة له من خلال تسريبها عبر قنوات تتعدى نتيجة الاستعلام. و من أهم هذه القنوات تسريب البيانات عبر التوابع المعرفة من قبل المستخدم user-defined functions. [5]

1-3-2-1- تسريب البيانات عبر التوابع المعرفة من قبل المستخدم :

فيما يلي مثالاً يوضح تسريب البيانات عبر التوابع : بفرض وجود سياسة أمنية تتحكم بالوصول إلى جدول الزبائن customers[1] بحيث تسمح للمستخدمين بالوصول إلى بيانات الزبائن الذكور فقط في حين تمنعهم من الاطلاع على أي تفاصيل تخص الزبائن الإناث ، و بفرض أن تابع السياسة الأمنية يأخذ الشكل :

```
create or replace function PF (schema in varchar2, tab in varchar2) return varchar2 as
begin
  return q'[s(gender)= 'true']';
end;
```

حيث **S** تابع بسيط يأخذ الصيغة التالية :

```
create or replace function S (gender varchar2) return varchar2 as
begin
```

[1] **يعتبر هذا الجدول جزءاً من OE schema وهي مجموعة من العلاقات خاصة للأغراض التدريبية طرحت من قبل Oracle DBMS .**

```
 if gender like 'm'
then return 'true';
 else return 'false';
end if;
 end;
```

و بفرض أن pcm مستخدم في قاعدة البيانات يملك سماحية الاستعلام من الجدول customers كما يملك أيضاً سماحية إنشاء تابع في قاعدة البيانات . و استخدم هذه السماحية في إنشاء التابع التالي :

```
create or replace function leak (email in varchar2,gender in varchar2, marital_status in varchar2) return varchar2 as
begin
 dbms_output.put_line('email ='||email||' marital_status ='||marital_status||' gender ='||gender);
 return 'true'; end;
```

و بعد ربط السياسة الأمنية الممثلة بالتابع PF مع الجدول customers نجد أن إصدار pcm للاستعلام التالي :

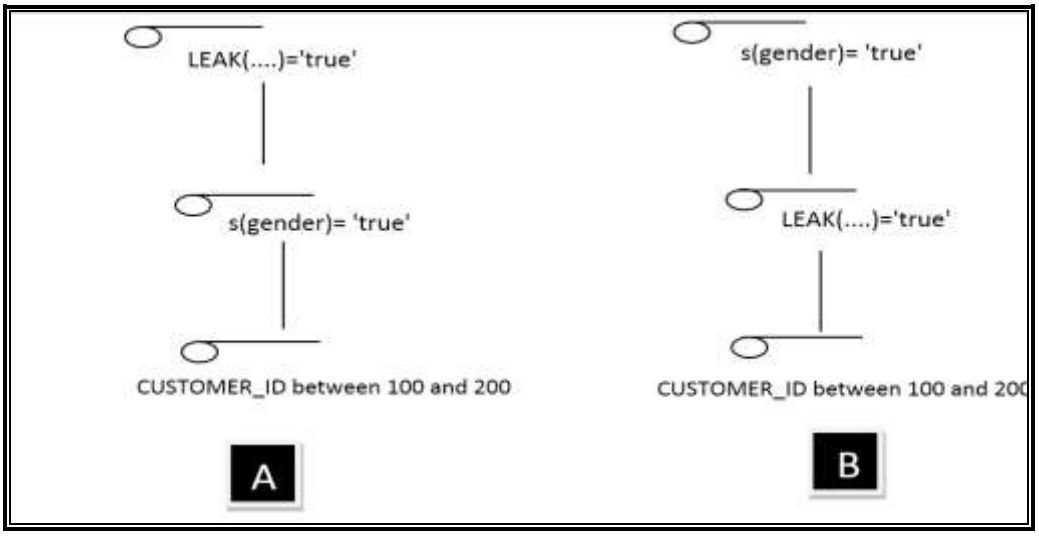
```
select cust_email,gender,marital_status
from oe.customers
where customer_id between 100 and 200
and leak(cust_email,gender,marital_status)='true';
```

سيؤدي بشكل آلي إلى استدعاء تابع السياسة المرافق (PF) ليقوم بتعديل الاستعلام السابق ليصبح بالشكل :

```
select cust_email,gender,marital_status from oe.customers
where customer_id between 100 and 200
and leak(cust_email,gender,marital_status)='true' and s(gender)= 'true';
```

وذلك لضمان وصول المستخدم للمعلومات المتاحة له فقط . نلاحظ من شكل الاستعلام المعدل أنه يحتوي على ثلاثة شروط يتحكم الـ optimizer في اختيار ترتيب تطبيقها على سجلات الجدول customers بما يضمن تنفيذ الاستعلام بفعالية . وبما أن أحد هذه الشروط يضم مفتاحاً رئيسياً (customer_id) فسيجري تطبيقه أولاً لكونه مفهرساً بشكل تلقائي ، وبالتالي يتبقى أمام optimizer خطتين محتملتين لتنفيذ الاستعلام هما : الخطتان A, B الموضحتان بالشكل(10).

إن أي خطة تنفيذ سيتم اختيارها ستعيد نفس النتائج للمستخدم و مع ذلك تعد الخطة A أكثر أمناً من الخطة B ، لكون سجلات الجدول customers سيتم تمريرها إلى التابع Leak بعد أن تكون قد خضعت لاختبار التخويل (s(gender)=true) . و بالتالي ليس هناك أي خطر مترتب على سلوك التابع إزاء هذه السجلات طالما أنها متاحة للمستخدم أصلاً .



الشكل(10): خطط التنفيذ المحتملة أمام optimizer

ولكن لو حصل و اختار الـ optimizer الخطة B فهذا يعني تمرير السجلات لتابع التسريب Leak قبل أن تخضع لاختبار التخويل ، وبما أن هذا التابع معرف من قبل المستخدم ولا يخضع سلوكه للسياسة الأمنية فإنه لا يمكن التحكم بما يمكن أن يفعله بالبيانات . وهنا تماماً يحدث التسريب .

اتضح لنا أثناء تنفيذ الاستعلام السابق أن optimizer سيختار الخطة B ، و بالتالي سيتم اختراق السياسة الأمنية التي تحمي الجدول customers و طباعة بيانات الزبائن الإناث على الشاشة . كما نرى من الشكل (11):

```
EMAIL =Constantin.Welles@ANHINGA.EXAMPLE.COM    MARITAL_STATUS =married    GENDER =M
EMAIL =Harrison.Pacino@ANI.EXAMPLE.COM    MARITAL_STATUS =single    GENDER =M
EMAIL =Manisha.Taylor@AUKLET.EXAMPLE.COM    MARITAL_STATUS =married    GENDER =F
EMAIL =Harrison.Sutherland@GODWIT.EXAMPLE.COM    MARITAL_STATUS =single    GENDER =F
EMAIL =Matthias.MacGraw@GOLDENEYE.EXAMPLE.COM    MARITAL_STATUS =married    GENDER =F
EMAIL =Matthias.Hannah@GREBE.EXAMPLE.COM    MARITAL_STATUS =single    GENDER =M
EMAIL =Matthias.Cruise@GROSBEAK.EXAMPLE.COM    MARITAL_STATUS =married    GENDER =F
EMAIL =Meenakshi.Mason@JACANA.EXAMPLE.COM    MARITAL_STATUS =married    GENDER =M
EMAIL =Christian.Cage@KINGLET.EXAMPLE.COM    MARITAL_STATUS =single    GENDER =M
EMAIL =Charlie.Sutherland@LIMPKIN.EXAMPLE.COM    MARITAL_STATUS =married    GENDER =M
EMAIL =Charlie.Pacino@LONGSPUR.EXAMPLE.COM    MARITAL_STATUS =single    GENDER =M
EMAIL =Guillaume.Jackson@MOORHEN.EXAMPLE.COM    MARITAL_STATUS =married    GENDER =M
```

الشكل(11): تسريب البيانات من خلال التابع المعرف من قبل المستخدم

علماً أن اختيار أي من الخطتين A,B لن يؤثر على نتيجة الاستعلام المعادة للمستخدم إذ ستتضمن السجلات الخاصة بالزبائن فقط كما يتضح من الشكل(12):

CUST_EMAIL	GENDER	MARITAL_STATUS
Markus.Rampling@PUFFIN.EXAMPLE.COM	M	single
Goldie.Slater@PYRRHULOXIA.EXAMPLE.COM	M	married
Divine.Aykroyd@REDSTART.EXAMPLE.COM	M	single
Dieter.Matthau@VERDIN.EXAMPLE.COM	M	married
Divine.Sheen@COWBIRD.EXAMPLE.COM	M	single
Frederico.Romero@CURLEW.EXAMPLE.COM	M	married
Sidney.Capshaw@DUNLIN.EXAMPLE.COM	M	single
Frederico.Lyon@FLICKER.EXAMPLE.COM	M	married
Eddie.Boyer@GALLINULE.EXAMPLE.COM	M	married
Eddie.Stern@GODWIT.EXAMPLE.COM	M	married

الشكل(12): نتائج الاستعلام في حال اختيار أي من الخطتين A,B

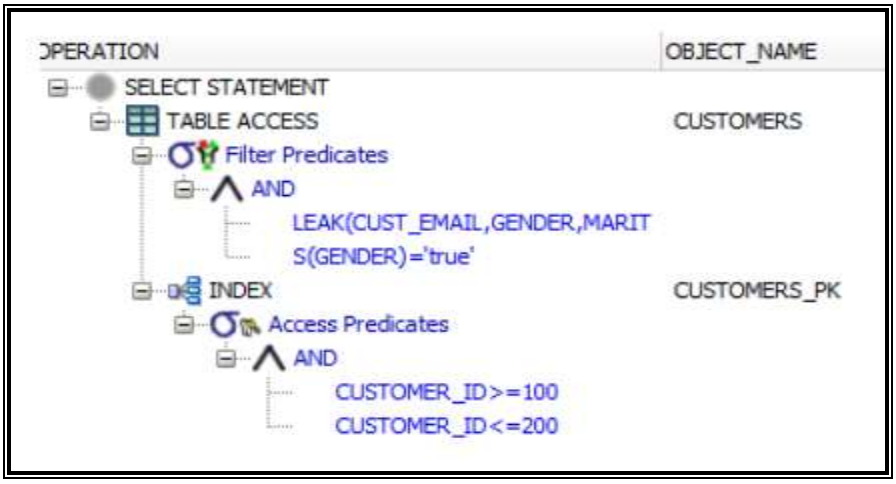
1-3-2-2- تفسير النتيجة واقتراح الحلول :

لتفسير النتيجة التي حصلنا عليها قمنا بداية بدراسة خطة تنفيذ الاستعلام المبينة بالشكل (13) وفيها نجد أن محرك SQL قام بداية بجلب السجلات التي تحقق شرط الـ customer_id لكونه مفتاحاً رئيسياً مفهرساً ، و بعد ذلك تم تنفيذ الشرط المحتوي على تابع التسريب Leak قبل تنفيذ تابع السياسة الأمنية مما أدى لتسريب البيانات الخاصة بالزبائن الإناث.

لعل أول تحليل يتبادر إلى الذهن لدى رؤية النتيجة أن اختيار الـ optimizer لتنفيذ الشرط الخاص بالتابع Leak أولاً يعود إلى أن تعقيده أقل من تعقيد تابع السياسة الأمنية . و لكن تبين لنا تجريبياً دحض هذه الفكرة إذ أن زيادة تعقيد التابع Leak لم تغير النتيجة السابقة و حصل تسريب بالبيانات .

مما جعلنا نبحث عن طريقة تمكننا من وضع التوابع المعرفة من قبل المستخدم في رأس أي خطة للتنفيذ . بمعنى أن يتم تطبيقها بعد كل الشروط الأخرى بما يضمن وصول بيانات متاحة فقط إلى هذه التوابع .

لذلك لجأنا لدراسة آلية تعامل الـ optimizer مع الشروط الواردة في عبارة where . ووجدنا أن الـ optimizer يقوم بعدة تحويلات على الاستعلام من أجل ايجاد الطريقة المثلى لتنفيذه بأقل تكلفة ، و من ضمن هذه التعديلات ما يعرف بـ view merging و predicate pushing و الفرق بينهما باختصار كما يلي :



الشكل(13): خطة تنفيذ الاستعلام B المسببة للتسريب

- ✓ view merging : يتم هنا دمج تعريف المنظار والشروط الواردة في الاستعلام المرافق له مع استعلام المستخدم الأصلي ، بهدف الاستغناء عن بعض العمليات المكررة بين المنظار و الاستعلام الأصلي .
- ✓ predicate pushing : هو النوع الثاني من التحويلات و يعمل بأسلوب معاكس للتحويل الأول إذ يقوم بدمج الشروط الواردة في استعلام المستخدم الاساسي مع الاستعلام المرافق لتعريف المنظار .[6]

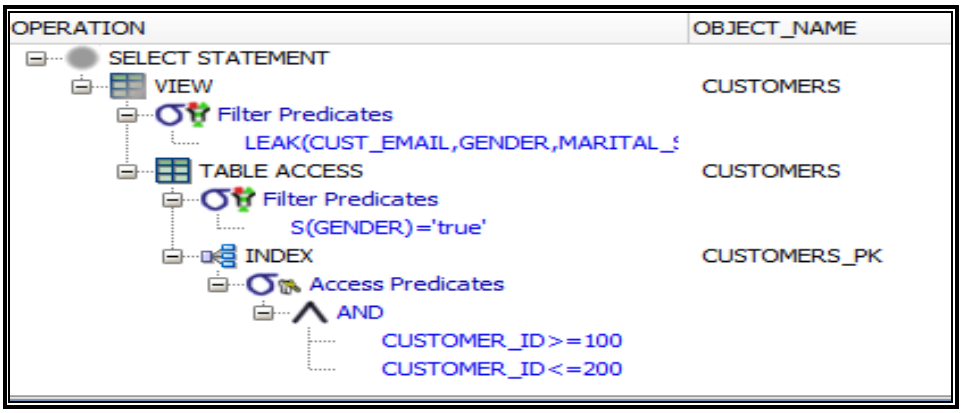
و إذا نظرنا إلى السياسة الأمنية و كأنها نافذة تقيد وصول المستخدم لسجلات و أعمدة محددة من الجداول الأساس فإنها تشبه في آلية عملها المنظار تماماً . وبالتالي عند منع عملية دمج الشروط سنتمكن من تطبيق شرط السياسة الأمنية بشكل مستقل عن الشروط الواردة في الاستعلام الأصلي ، و بالتالي سنضمن أن ما يتم تمريره إلى التابع المستخدم للتجسس Leak هو فقط الأسطر المسموحة من قبل تابع السياسة الأمنية .

و هذا ما قمنا به تماماً لنجد النتيجة رائعة بمنع تسريب البيانات عبر تابع التجسس وطباعة بيانات الزبائن فقط على شاشة المستخدم معرف التابع Leak كما يتضح من الشكل (14):

```
EMAIL =Constantin.Welles@ANHINGA.EXAMPLE.COM    MARITAL_STATUS =married    GENDER =M
EMAIL =Harrison.Pacino@ANI.EXAMPLE.COM    MARITAL_STATUS =single    GENDER =M
EMAIL =Matthias.Hannah@GREBE.EXAMPLE.COM    MARITAL_STATUS =single    GENDER =M
EMAIL =Meenakshi.Mason@JACANA.EXAMPLE.COM    MARITAL_STATUS =married    GENDER =M
EMAIL =Christian.Cage@KINGLET.EXAMPLE.COM    MARITAL_STATUS =single    GENDER =M
EMAIL =Charlie.Sutherland@LIMPKIN.EXAMPLE.COM    MARITAL_STATUS =married    GENDER =M
EMAIL =Charlie.Pacino@LONGSPUR.EXAMPLE.COM    MARITAL_STATUS =single    GENDER =M
EMAIL =Guillaume.Jackson@MOORHEN.EXAMPLE.COM    MARITAL_STATUS =married    GENDER =M
EMAIL =Daniel.Costner@PARULA.EXAMPLE.COM    MARITAL_STATUS =single    GENDER =M
EMAIL =Dianne.Derek@SAW-WHET.EXAMPLE.COM    MARITAL_STATUS =married    GENDER =M
EMAIL =Geraldine.Schneider@SCAUP.EXAMPLE.COM    MARITAL_STATUS =married    GENDER =M
EMAIL =Geraldine.Martin@SCOTER.EXAMPLE.COM    MARITAL_STATUS =single    GENDER =M
```

الشكل(14): ناتج تابع التسريب بعد منع عملية دمج الشروط

و بدراسة خطة تنفيذ الاستعلام بعد منع عمليات دمج الشروط الموضحة في الشكل(15) نجد أن شرط تابع السياسة الأمنية قد تم تنفيذه قبل الشرط الخاص بتابع التسريب مما ضمن تمرير سجلات مسموحة فقط للتابع Leak .



الشكل(15) : منع التسريب باستخدام الخطةA

الاستنتاجات و التوصيات :

1- يعتبر النموذج Non-Truman خياراً جيداً في التطبيقات التي تتعامل مع عدد محدود من المستخدمين و تكون بياناتها على درجة عالية جداً من الحساسية بينما يمكن تطبيق النموذج Truman بشكل أوسع.

2- إن التحسن الكبير في الأداء الذي ينتج عن استخدام السياسات الساكنة يدفعنا كمطورين إلى العمل على بنية الشرط بحد ذاته لجعله قابلاً للتخزين ، مما يقلل من عدد مرات تنفيذ تابع السياسة الأمنية ، و بالتالي التخلص من العبء الذي يسببه في بطء الاستجابة لطلبات المستخدمين. و يصبح الشرط قابلاً للتخزين عندما نبتعد في صياغته عن القيم الثابتة و نستبدلها بواصفات سياق التطبيق ولكن مع ضمان تهيئتها بطريقة سليمة و آمنة .

3- تعتبر واصفات السياق مفتاح الوصول إلى البيانات لذلك يجب ضمان عملية التهيئة الآمنة لها ، والاعتماد في التهيئة على مدخلات المستخدمين يعتبر ثغرة أمنية كبيرة أشبه ما تكون بمن يبني سوراً منيعاً و يترك أبوابه مفتوحة .

4- يفيد application context بشكل خاص عندما تعتمد عملية الوصول إلى البيانات على أكثر من واصفة ، فبدلاً من تكرار تنفيذ عدة استعلامات جزئية متتالية لاسترجاع قيم هذه الواصفات من مخدمات قاعدة البيانات التي يمكن أن تكون بعيدة ، فإن تخزينها ضمن application context يؤدي إلى سرعة كبيرة في استرجاعها وبالتالي تحسناً كبيراً في الأداء .

5- يمكن لتخزين الشرط أن يكون على مستوى الجلسات sessions و الجداول و الهيكليات كذلك مما يضاعف من قدرته على تحسين سرعة الاستجابة لطلبات المستخدمين ويعتبر هذا مفيداً جداً في البيئات التي تضم عدد كبير من المستخدمين يصدرون الاستعلامات نفسها بشكل متكرر . وتتضاعف هذه الفائدة مع زيادة عدد الجداول المحمية بالسياسة نفسها.

6- يمكن للسياسات الستاتيكية المتأثرة بتغييرات سياق التنفيذ أن تكون خياراً أمثل للمدير الأمني في الحالات التي تتنوع فيها الشروط بتنوع المستخدمين أو الأدوار التي يؤدونها في المنظمة .

7- يعمل تخزين الشرط بفعالية عالية عندما يكون شرط الوصول إلى البيانات ثابت بينما تتغير قيم واصفات الوصول التي يعتمد عليها الشرط و يجب ألا نغفل عن بعض الحالات التي نضطر فيها إلى التوليد الديناميكي للشرط لضمان أمن البيانات .

8- إن تنفيذ تابع السياسة الأمنية قبل تنفيذ أي تابع آخر يرد في شروط الاستعلام يمكنه أن يحمي البيانات من التسريب لأنه يضمن تمرير بيانات متاحة فقط للتابع حتى لو كان خارجاً عن سيطرة سياسة التحكم بالوصول.

المراجع :

[1] RIZVI, S.; MENDELZON, A. ; SUDARSHAN, S. ; & ROY, P. *Extending query rewriting techniques for fine-grained access control*. In Proceedings of the 2004 ACM SIGMOD international conference on Management of data. (2004, June). (pp. 551562).

[2] WANG, Q. ; YU, T.; LI, N.; LOBO, J.; BERTINO, E.; IRWIN, K., & BYUN, J. W. *On the correctness criteria of fine-grained access control in relational databases.* In Proceedings of the 33rd international conference on Very large databases, (2007, September). (pp. 555-566). VLDB Endowment.

[3] SHETE, S. S., & KULKARNI, C. S. *Database Security using Role-Based Access Control System*. International Journal of Engineering Science, 8047. 2016

[4] OPYRCHAL, L.; COOPER, J.; POYAR, R.; LENAHAN, B.; & ZEINNER, D. *Bouncer: Policy-based fine grained access control in large databases*. International Journal of Security and Its Applications, 5**(2)**, 2011, 1-16.

[5] SUDERSHAN, S.; KABRA, G.; & RAMAMURTHY, R. *Redundancy and Information Leakage in Fine-Grained Access Control*. In *ACM SIGMOD*. 2006.

[6] GREEN, C. D. *Oracle9i Database Performance Tuning Guide and Reference*, Release 2 (9.2) Part No. A96533-02.

[7] ROLAND ,S. *Creating a personal view on the data via using a Virtual Private Database.* Business & Decision Life Sciences. Belgium: Brussels,2014

[8] Burleson,D. *Oracle hard-parse vs. soft parse*.Accessed in 20/9/2017. available at http://www.dbaoracle.com/t_hard_vs_soft_parse_parsing.htm

[9] Dhage, V. N., & Shelke, R. R. (2012). *Analysis of Fine-Grained Access Control in Database. International Journal of Computer Applications*, 4, 19-21.

[10] Sehta N., Jain S., (2012) . *A Fine Grained Access Control Model for Relational Databases*. (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 3 (1), 3183 – 3186.