\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |
| --- | --- |
|  | ИНСТИТУТ ИНТЕЛЛЕКТУАЛЬНЫХ КИБЕРНЕТИЧЕСКИХ СИСТЕМ  Кафедра №42  «Криптология и кибербезопасность» |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Федеральное государственное автономное образовательное учреждение высшего образования*

**«Национальный исследовательский ядерный университет «МИФИ»»**

ЛАБОРАТОРНАЯ РАБОТА №2-5:

**«Аудит БД»**

Аверин Владислав

Группа: Б19-505

Декабрь, 2022

*Содержание*

[*1.* *Политики аудита* 4](#_Toc121075841)

[*2.* *AUDIT\_ADMIN и AUDIT\_VIEWER* 7](#_Toc121075842)

[*3.* *Проверка работоспособности* 8](#_Toc121075843)

[*4.* *Очистка журнала* 10](#_Toc121075844)

[*Выводы:* 13](#_Toc121075845)

*Цель работы*

Приобрести базовые навыки проведения аудита в базе данных и анализа журнала аудита.

*Ход работы*

1. Разработать политику аудита для разрабатываемой легенды: какие действия пользователей (использование привилегий или ролей, в частности: использование системных привилегий для создания объектов) и какие объекты схемы данных требуют ведения журнала аудита?
2. Присвоить привилегии AUDIT ADMIN и AUDIT VIEWER соответствующим пользователям/ролям;
3. Использовать инструкцию CREATE AUDIT POLICY для реализации одной или нескольких политик аудита. Использовать инструкцию AUDIT для включения этих политик. Политики должны включать хотя бы по одному случаю аудита: системных привилегий, ролей, объектов;
4. Совершить несколько действий от лица контролируемых пользователей, либо над контролируемыми объектами;
5. От лица уполномоченного пользователя выполнить запрос к представлению журнала аудита (UNIFIED AUDIT TRAIL), убедиться в успешной регистрации событий аудита, изучить представленную в журнале информацию;
6. Очистить (PURGE) журнал аудита;
7. Оформить отчёт.

# *Политики аудита*

Так как политики аудита в Oracle Database могут быть направлены как на всех, так и на конкретных пользователей/роли, то для реализации аудита общего направления включим встроенные единые политики (Unified Audit Policy), такие как:

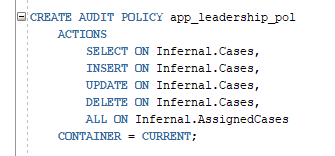
* *Logon Failures Predefined Unified Audit Policy* *(ORA\_LOGON\_FAILURES)* – политика аудита неудачных входов в систему (отслеживает только неудачные попытки);
* *Secure Options Predefined Unified Audit Policy (ORA\_SECURECONFIG)* – политика аудита “безопасной конфигурации” настроек (хз, как нормально перевести): включает аудит всех типичных действий, требующих различных привилегий для работы с объектами базы данных (не считая DML выражений);
* *User Account and Privilege Management Predefined Unified Audit Policy (ORA\_ACCOUNT\_MGMT)* – аудит выдачи и отзыва привилегий (GRANT и REVOKE), а также действий, связанных с изменением учетных записей (пользователей и ролей).

Эти стандартные политики охватывают, пожалуй, подавляющее большинство действий, которые вообще могут выполняться пользователями с какими-либо привилегиями. Однако в разработанной базе данных ни у кого, кроме локального и sys администраторов, в принципе нет таких прав (т.е. если мы видим какие-нибудь GRANT ANY PRIVILEGE или CREATE ROLE у условно бухгалтера, то это уже аларм). Более того, среди этих и других предоставленных по умолчанию политик не было какой-либо, учитывающей DML выражения и действия над контролируемыми таблицами (e.g. ALTER/DROP TABLE). Все потому, что данные категории действий относят к *Individually Auditing Schema Objects* – Индивидуальному аудиту объектов схемы пользователей (<https://docs.oracle.com/cd/E11882_01/server.112/e10575/tdpsg_auditing.htm#TDPSG50526:~:text=a%20multitier%20environment-,Individually%20Auditing%20Schema%20Objects,-Schema%20object%20auditing>)

Именно настройка такого типа политик будет проведена для нескольких наиболее используемых ролей в разрабатываемой БД, чтобы отслеживать изменения именно данных, т.к использование системных привилегий уже будут контролировать единые политики аудита. Рассмотрим роли глав отделов и бухгалтерии.

Главы отделов.

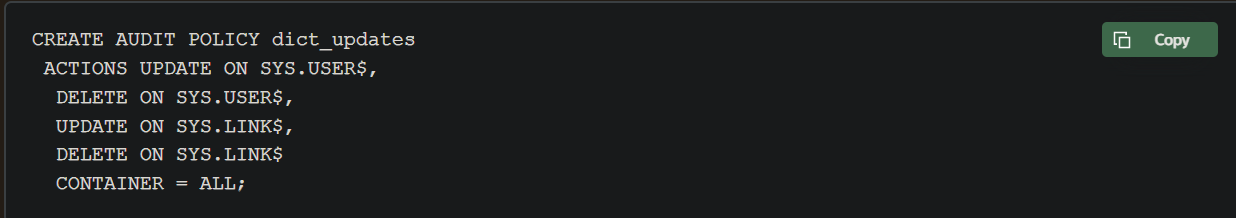
Напомню, за что по моей легенде отвечает данная роль. Они манипулируют таблицами Cases и AssignedCases, выдавая/изменяя дела своим подчиненным; поэтому логично было бы реализовать аудит данных таблиц для рассматриваемой роли, а именно:



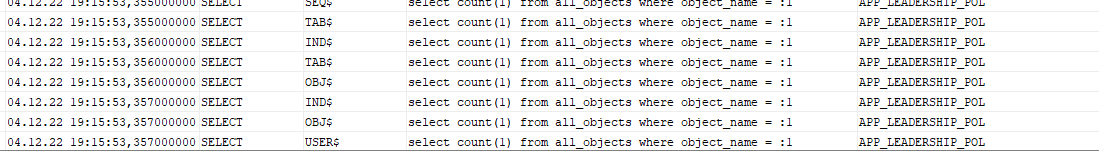


(Current при подключении через sqldeveloper является нашим контейнером: )

Alarm: я не знаю, почему так, но перечисление нескольких действий над таблицей через запятую он воспринимает как сепарабельные действия; т.е. в ACTIONS SELECT, *INSERT ON table* фраза ON будет относиться только к INSERT:

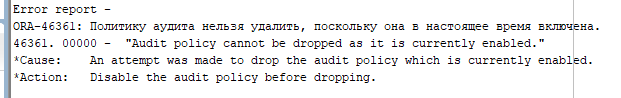


Ибо пока я этого не заметил, практически на каждый запрос СУБД записывала 100500 (примерно по десятку) DML записей в журнал аудита:



Поэтому это не так работает, для каждого действия задаем отдельно объект. Либо как я сделал для наглядности с AssignedCases, сразу пишем *ACTIONS ALL … EXCEPT.*

Alarm x2: если отключить политику, включенную конкретно пользователю или роли, то просто *NOAUDIT POLICY policy\_name;* не прокатит. Отключение политик должно быть таким же, как включение, с заменой первого слова AUDIT на NOAUDIT (а то я слишком много времени потратил на то, чтобы понять, почему он ругался:



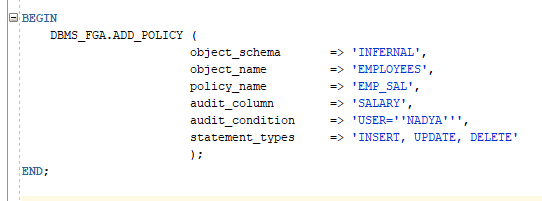
Теперь нужно включить политику для нашей роли. В отличие от перечисления пользователей, которым достаточно ключевого слова *BY*, для использования ролей надо использовать *BY USERS WITH GRANTED ROLES*:





Это все, конечно, хорошо, и теперь можно отслеживать действия, производимые данной ролью над перечисленными таблицами. Однако такой подход ограничивается такой унитарной единицей, как таблица/роль/представление. Если мы хотим спуститься до уровня элементов таблиц (а мы как бы хотим), то лучшим вариантом будет использование детализированного аудита (пакета DBMS\_FGA), который позволяет формировать политики на более низком уровне. Политики, созданные посредством этого пакета, управляются исключительно через него же, но могут применяться ко всем командам DML, т.е. мониторить изменения конкретных столбцов и даже выполнять аудит по конкретному значению атрибута, что более удобно (и привычно, т.к. пакет по сути очень похож просто на интерфейс с методами и параметрами как в других ЯП).

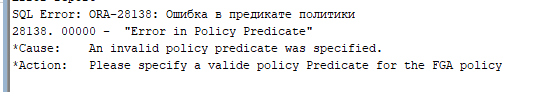
В прошлых лабораторных таблица Employees была немного модифицирована: был добавлен столбец salary, за который отвечают бухгалтера. Реализуем аудит роли бухгалтерии по этому столбцу при помощи детализированного аудита DBMS\_FGA:





Теперь всякий раз, когда пользователь Nadya прямо или косвенно будет использовать данные в поле ‘SALARY’, события об этом будут записываться в унифицированный журнал аудита.

Note: в отличие от CREATE AUDIT POLICY у меня не получилось задать политику для роли, на audit\_condition ROLE=accounting\_role он ругался:



У меня то и USER= сработал чисто методом тыка :) Поэтому пришлось оставить с USER=Nadya. И вообще напрягает, что возможность реализации параметров политики в FGA происходит уже при ее срабатывании, т.е. какие-то пользователи при неправильной настройке технически могут столкнуться с тем, что изображено на скрине выше. В то время как синтаксис унифицированной политики проверяется сразу на этапе создания (по идее, это из-за принципа работы: унифицированная политика создается сразу как объект, а FGA по сути будет являться процедурой, которая выполняется при срабатывании триггера, но это чисто догадки).

# *AUDIT\_ADMIN и AUDIT\_VIEWER*

В случае рассматриваемой БД мне кажется нецелесообразным давать каким-то другим ролям, помимо созданного администратора БД доступ к журналу аудита – им он просто не нужен (все созданные ранее роли относятся к функционированию самого субъекта БД – полицейского участка, а не непосредственно к системному управлению БД). Поэтому привилегия AUDIT\_ADMIN будет у sys администратора (я ее выдал уже после того, как подумал, есть ли эта привилегия по умолчанию, поэтому не знаю, необходимо ли было ее выдавать), а AUDIT\_VIEWER выдадим локальному администратору Infernal:





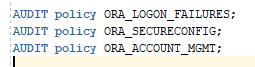




# *Проверка работоспособности*

Первое, что нужно бы отметить, это то, что если политика включается во время работы конкретной схемы/роли, то пользователь должен обязательно перезайти, иначе политика для него не начнет работать.

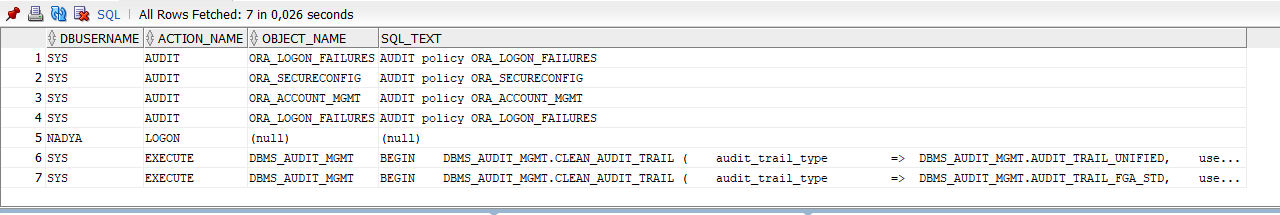
Второе, включаем унифицированные политики. Все 3 политики (*ORA\_LOGON\_FAILURES, ORA\_SECURECONFIG и ORA\_ACCOUNT\_MGMT)* уже есть в списке политик, поэтому их необходимо просто активировать:





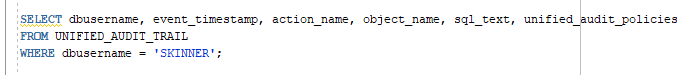
Проверку того, что они начали работать, можно провести уже сразу:

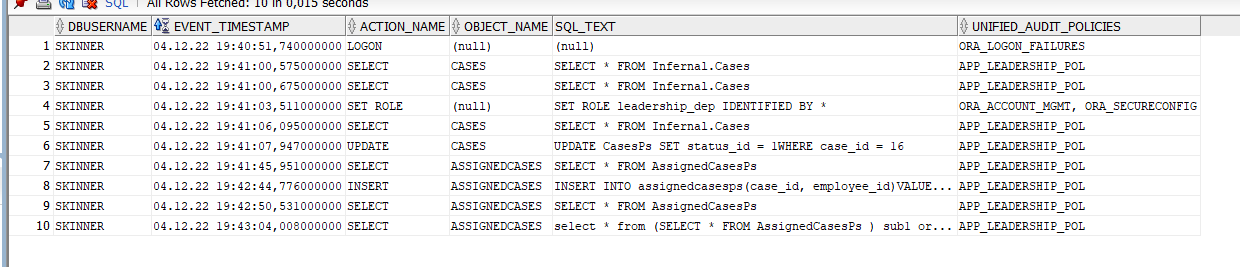




Теперь перейдем к рукописным политикам.

Для Скиннера (главы отдела):

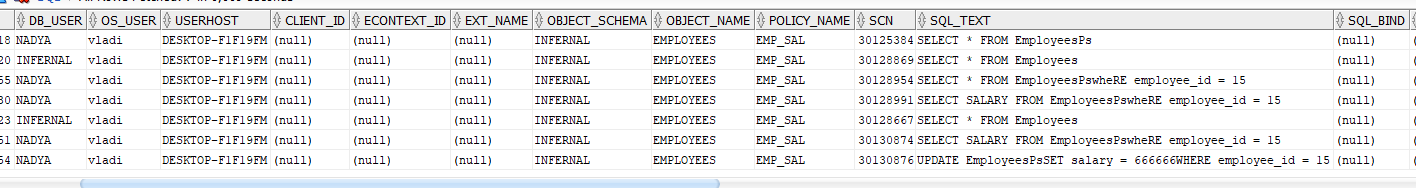




Как видно, производится аудит не только по встроенным включенным политикам ORA\_, но и наша app\_leadership\_pol.

Для FGA политик есть свой отдельный журнал, DBA\_FGA\_AUDIT\_TRAIL:





(Запрос на выборку SELECT \* FROM EmployeesPs там появился до того, как я добавил statement\_types)

Но вот данные по привилегиям, когда, например, я выдавал Наде роль при перезаходе, записывается в унифицированный журнал аудита (и пароль, кстати благоразумно не палится):

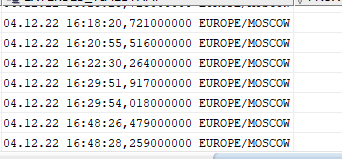




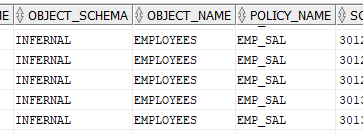
Поля, которые предоставляются в журналах, немного отличаются, то я выделю некоторые особо полезные для выборок данных.

FGA-журнал:

* Timestamp/extended timestamp: второй вариант помимо времени выводит доли секунды (или что это) и геолокацию:



* db\_user – имя схемы, которая спровоцировала вызов аудита;
* os\_user, userhost – системные атрибуты;
* object\_schema, object\_name, policy\_name – переменные, которые мы вводили при создании FGA политики:



* SQL\_text – текст запроса;
* statement type – SELECT, UPDATE, INSERT, SELETE, etc.;
* current\_user – если честно, я не особо понял разницы от db\_user.

В UNIFIED\_AUDIT журнале к этим полям добавляются расширенные:

* system\_privilage\_used – использовались ли привилегии определенного уровня: полезно, например, просто выполнить выборку по этому атрибуту чтобы увидеть, кто пользовался привилегиями sysdba;
* role – роль пользователя dbusername (аналог db\_user из FGA-журнала);
* unified\_policy\_name – понятно из названия;
* fga\_policy\_name – вот это немного непонятно, чему там, кроме null, может быть равно.

А вообще, по сравнению с FGA-журналом, где всего 28 столбцов, в унифицированном журнале их около сотни, что “немного” больше. Так что там можно найти достаточно много информации по каждому инциденту.

# *Очистка журнала*

Насколько я понял, самый быстрый способ вручную очистить UNIFIED\_AUDIT\_TRAIL, это воспользоваться еще одним пакетом (менеджера записей журнала аудита) DBMS\_AUDIT\_MGMT:

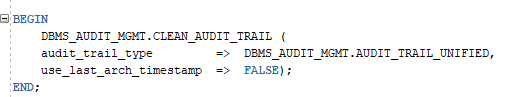
*BEGIN*

*DBMS\_AUDIT\_MGMT.CLEAN\_AUDIT\_TRAIL (*

*audit\_trail\_type => DBMS\_AUDIT\_MGMT.AUDIT\_TRAIL\_UNIFIED,*

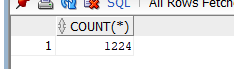
*use\_last\_arch\_timestamp => FALSE);*

*END;*

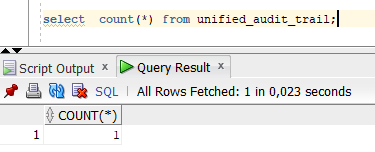


Функция очистит полностью весь унифицированный журнал, не учитывая *last archive timestamp* (как я понял, этот параметр связан с другим - audit\_trail\_purge\_interval, который указывает интервал в часах, как часто очищать журнал аудита. То есть если use\_last\_arch\_timestamp указан в TRUE, как он стоит по умолчанию, то очищаться будут все записи, старше указанного интервала времени. При значении FALSE раз в интервал будут удаляться вообще все записи). А так как мы никакого интервала не указываем, то параметр должен иметь значение FALSE (иначе никакие записи не удалятся).

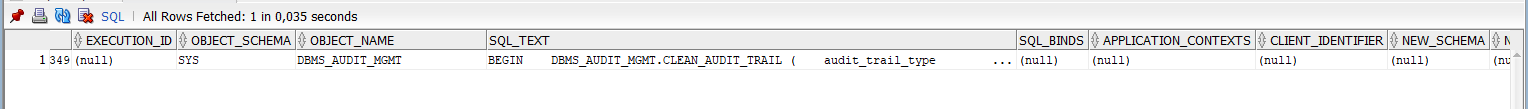
До:



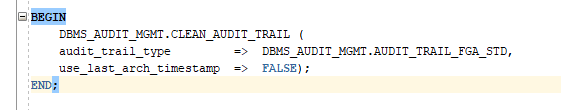
После:



Эта одна запись про очистку журнала (хе-хе):



Для очистки FGA журнала воспользуемся другой компонентой AUDIT\_TRAIL\_FGA\_STD:

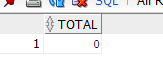


До:





После:



Вообще говоря, есть параметр AUDIT\_TRAIL\_ALL для всех типов аудитов (как SYS.AUD$, SYS.FGA\_LOG$ и унифицированного аудита так и аудита операционной системы и XML), но я так и не понял, можно ли его использовать, поломаю еще что-то.

# *Выводы:*

В результате данной лабораторной работы была изучена механика работы журнала аудита, возможности его конфигурации и мониторинга. Установлены некоторые встроенные единые политики, написаны две индивидуальных для двух разных ролей, протестирована их работоспособность от лица ADMIN\_VIEWER пользователя, а также было произведено поверхностное знакомство с пакетом DBMS\_AUDIT\_MGMT.

Что интересно. Включение стандартных политик уже производит запись десятков различных событий для выполнения простого запроса от лица какой-либо роли. Наверняка объем журнала (предельное значение которого, кстати, можно задавать для автоматической очистки) не такой уж и большой. И при интенсивном параллельном использовании одного и того же контейнера наверняка он заполнится быстро. Что подводит нас к проблеме логирования, которая есть и в компьютерных сетях. С учетом зеркалирования и необходимого времени жизни логов, необходимо найти баланс между отслеживаемыми действиями и поголовной записью всех событий. Поэтому скорее всего политики, которые реализованы здесь, достаточно неудачные: намного лучше было бы создать аудит неудачных действий, как с попытками входа (реализуется через WHENEVER NOT SUCCESSFUL), т.к. они в большей мере могут сказать о несанкционированном поведении, а занимать места в журнале будет намного меньше. Но это чисто догадки :)