



## **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
2. AUDIT_ADMIN и AUDIT_VIEWER .....	7
3. Проверка работоспособности .....	8
4. Очистка журнала .....	10
Выводы: .....	13

## *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 контролируруемыми 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.

## 1. 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 контролируруемыми таблицами (е.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#TDP50526:~:text=a%20multitier%20environment-,Individually%20Auditing%20Schema%20Objects,-Schema%20object%20auditing](https://docs.oracle.com/cd/E11882_01/server.112/e10575/tdpsg_auditing.htm#TDP50526:~: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:

```
CREATE AUDIT POLICY app_leadership_pol
ACTIONS
  SELECT ON Infernal.Cases,
  INSERT ON Infernal.Cases,
  UPDATE ON Infernal.Cases,
  DELETE ON Infernal.Cases,
  ALL ON Infernal.AssignedCases
CONTAINER = CURRENT;
```

Audit POLICY created.

(Current при подключении через sqldeveloper является нашим контейнером: 

CON_NAME
XEFDDB1

 )

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:

```
CREATE AUDIT POLICY dict_updates
ACTIONS UPDATE ON SYS.USER$,
  DELETE ON SYS.USER$,
  UPDATE ON SYS.LINK$,
  DELETE ON SYS.LINK$
CONTAINER = ALL;
```

Copy

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

04.12.22 19:15:53,355000000	SELECT	select count(1) from all_objects where object_name = :1	APP_LEADERSHIP_POL
04.12.22 19:15:53,355000000	SELECT	select count(1) from all_objects where object_name = :1	APP_LEADERSHIP_POL
04.12.22 19:15:53,356000000	SELECT	select count(1) from all_objects where object_name = :1	APP_LEADERSHIP_POL
04.12.22 19:15:53,356000000	SELECT	select count(1) from all_objects where object_name = :1	APP_LEADERSHIP_POL
04.12.22 19:15:53,357000000	SELECT	select count(1) from all_objects where object_name = :1	APP_LEADERSHIP_POL
04.12.22 19:15:53,357000000	SELECT	select count(1) from all_objects where object_name = :1	APP_LEADERSHIP_POL
04.12.22 19:15:53,357000000	SELECT	select count(1) from all_objects where object_name = :1	APP_LEADERSHIP_POL
04.12.22 19:15:53,357000000	SELECT	select count(1) from all_objects where object_name = :1	APP_LEADERSHIP_POL

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:

Error report -

```
ORA-46361: Политику аудита нельзя удалить, поскольку она в настоящее время включена.  
46361. 00000 - "Audit policy cannot be dropped as it is currently enabled."  
*Cause:      An attempt was made to drop the audit policy which is currently enabled.  
*Action:     Disable the audit policy before dropping.
```

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*:

```
AUDIT policy app_leadership_pol BY USERS WITH GRANTED ROLES leadership_dep;
```

Audit succeeded.

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:

```
BEGIN  
  DBMS_FGA.ADD_POLICY (  
    object_schema      => 'INFERNAL',  
    object_name        => 'EMPLOYEES',  
    policy_name        => 'EMP_SAL',  
    audit_column       => 'SALARY',  
    audit_condition    => 'USER='NADYA'',  
    statement_types    => 'INSERT, UPDATE, DELETE'  
  );  
END;
```

PL/SQL procedure successfully completed.

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:

```
SQL Error: ORA-28138: Ошибка в предикате политики
28138. 00000 - "Error in Policy Predicate"
*Cause:      An invalid policy predicate was specified.
*Action:     Please specify a valide policy Predicate for the FGA policy
```

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).

## 2. 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:

```
GRANT AUDIT_ADMIN TO sys;
```

```
Grant succeeded.
```

```
GRANT AUDIT_VIEWER TO Infernal;
```

```
Grant succeeded.
```

### 3. 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:

```
AUDIT policy ORA_LOGON_FAILURES;  
AUDIT policy ORA_SECURECONFIG;  
AUDIT policy ORA_ACCOUNT_MGMT;  
|
```

```
Audit succeeded.
```

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

```
SELECT dbusername, action_name, object_name, sql_text  
FROM UNIFIED_AUDIT_TRAIL;
```

SQL   All Rows Fetched: 7 in 0,026 seconds				
DBUSERNAME	ACTION_NAME	OBJECT_NAME	SQL_TEXT	
1 SYS	AUDIT	ORA_LOGON_FAILURES	AUDIT policy ORA_LOGON_FAILURES	
2 SYS	AUDIT	ORA_SECURECONFIG	AUDIT policy ORA_SECURECONFIG	
3 SYS	AUDIT	ORA_ACCOUNT_MGMT	AUDIT policy ORA_ACCOUNT_MGMT	
4 SYS	AUDIT	ORA_LOGON_FAILURES	AUDIT policy ORA_LOGON_FAILURES	
5 NADYA	LOGON	(null)	(null)	
6 SYS	EXECUTE	DBMS_AUDIT_MGMT	BEGIN DBMS_AUDIT_MGMT.CLEAN_AUDIT_TRAIL ( audit_trail_type => DBMS_AUDIT_MGMT.AUDIT_TRAIL_UNIFIED, use...	
7 SYS	EXECUTE	DBMS_AUDIT_MGMT	BEGIN DBMS_AUDIT_MGMT.CLEAN_AUDIT_TRAIL ( audit_trail_type => DBMS_AUDIT_MGMT.AUDIT_TRAIL_FGA_STD, use...	

Теперь перейдем к рукописным политикам.

Для Скиннера (главы отдела):

```
SELECT dbusername, event_timestamp, action_name, object_name, sql_text, unified_audit_policies  
FROM UNIFIED_AUDIT_TRAIL  
WHERE dbusername = 'SKINNER';
```

SQL   All Rows Fetched: 10 in 0,015 seconds						
DBUSERNAME	EVENT_TIMESTAMP	ACTION_NAME	OBJECT_NAME	SQL_TEXT	UNIFIED_AUDIT_POLICIES	
1 SKINNER	04.12.22 19:40:51,740000000	LOGON	(null)	(null)	ORA_LOGON_FAILURES	
2 SKINNER	04.12.22 19:41:00,575000000	SELECT	CASES	SELECT * FROM Infernal.Cases	APP_LEADERSHIP_POL	
3 SKINNER	04.12.22 19:41:00,675000000	SELECT	CASES	SELECT * FROM Infernal.Cases	APP_LEADERSHIP_POL	
4 SKINNER	04.12.22 19:41:03,511000000	SET ROLE	(null)	SET ROLE leadership_dep IDENTIFIED BY *	ORA_ACCOUNT_MGMT, ORA_SECURECONFIG	
5 SKINNER	04.12.22 19:41:06,095000000	SELECT	CASES	SELECT * FROM Infernal.Cases	APP_LEADERSHIP_POL	
6 SKINNER	04.12.22 19:41:07,947000000	UPDATE	CASES	UPDATE CasesPs SET status_id = 1 WHERE case_id = 16	APP_LEADERSHIP_POL	
7 SKINNER	04.12.22 19:41:45,951000000	SELECT	ASSIGNEDCASES	SELECT * FROM AssignedCasesPs	APP_LEADERSHIP_POL	
8 SKINNER	04.12.22 19:42:44,776000000	INSERT	ASSIGNEDCASES	INSERT INTO assignedcasesps(case_id, employee_id)VALUE...	APP_LEADERSHIP_POL	
9 SKINNER	04.12.22 19:42:50,531000000	SELECT	ASSIGNEDCASES	SELECT * FROM AssignedCasesPs	APP_LEADERSHIP_POL	
10 SKINNER	04.12.22 19:43:04,008000000	SELECT	ASSIGNEDCASES	select * from (SELECT * FROM AssignedCasesPs ) sub1 or...	APP_LEADERSHIP_POL	



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:

```
SELECT * FROM dba_fga_audit_trail;
```

DB_USER	OS_USER	USERHOST	CLIENT_ID	ECONTEXT_ID	EXT_NAME	OBJECT_SCHEMA	OBJECT_NAME	POLICY_NAME	SCN	SQL_TEXT	SQL_BIND
NADYA	vladi	DESKTOP-F1F19FM	(null)	(null)	(null)	INFERNAL	EMPLOYEES	EMP_SAL	30125384	SELECT * FROM EmployeesPs	(null)
INFERNAL	vladi	DESKTOP-F1F19FM	(null)	(null)	(null)	INFERNAL	EMPLOYEES	EMP_SAL	30128869	SELECT * FROM Employees	(null)
NADYA	vladi	DESKTOP-F1F19FM	(null)	(null)	(null)	INFERNAL	EMPLOYEES	EMP_SAL	30128954	SELECT * FROM EmployeesPawhere employee_id = 15	(null)
NADYA	vladi	DESKTOP-F1F19FM	(null)	(null)	(null)	INFERNAL	EMPLOYEES	EMP_SAL	30128991	SELECT SALARY FROM EmployeesPawhere employee_id = 15	(null)
INFERNAL	vladi	DESKTOP-F1F19FM	(null)	(null)	(null)	INFERNAL	EMPLOYEES	EMP_SAL	30128667	SELECT * FROM Employees	(null)
NADYA	vladi	DESKTOP-F1F19FM	(null)	(null)	(null)	INFERNAL	EMPLOYEES	EMP_SAL	30130874	SELECT SALARY FROM EmployeesPawhere employee_id = 15	(null)
NADYA	vladi	DESKTOP-F1F19FM	(null)	(null)	(null)	INFERNAL	EMPLOYEES	EMP_SAL	30130876	UPDATE EmployeesPs SET salary = 666666WHERE employee_id = 15	(null)

(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):

```
SELECT * FROM UNIFIED_AUDIT_TRAIL;
```

10 4,959000000 EXECUTE	0 7544:20348	0000000000000000	30130833 (null)	SYS	DBMS_FGA	DECLARE SqlDevBindI2_1 VARCHAR2(32767):=:SqlDevBindI2Init1;
11 4,490000000 SET ROLE	0 7544:20520	(null)	30130840 (null)	(null)	(null)	SET ROLE ACCOUNTING_dep IDENTIFIED BY *

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:

04.12.22 16:18:20,721000000	EUROPE/MOSCOW
04.12.22 16:20:55,516000000	EUROPE/MOSCOW
04.12.22 16:22:30,264000000	EUROPE/MOSCOW
04.12.22 16:29:51,917000000	EUROPE/MOSCOW
04.12.22 16:29:54,018000000	EUROPE/MOSCOW
04.12.22 16:48:26,479000000	EUROPE/MOSCOW
04.12.22 16:48:28,259000000	EUROPE/MOSCOW

- 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:

IE	OBJECT_SCHEMA	OBJECT_NAME	POLICY_NAME	SC
	INFERNAL	EMPLOYEES	EMP_SAL	301
	INFERNAL	EMPLOYEES	EMP_SAL	301
	INFERNAL	EMPLOYEES	EMP_SAL	301
	INFERNAL	EMPLOYEES	EMP_SAL	301
	INFERNAL	EMPLOYEES	EMP_SAL	301

- 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\_privilege\_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.

## 4. 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,
```

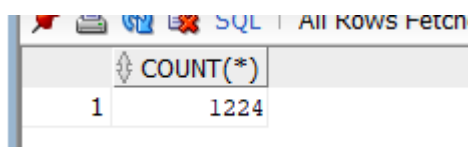
```
use_last_arch_timestamp => FALSE);
```

END;

```
BEGIN
  DBMS_AUDIT_MGMT.CLEAN_AUDIT_TRAIL (
    audit_trail_type      => DBMS_AUDIT_MGMT.AUDIT_TRAIL_UNIFIED,
    use_last_arch_timestamp => FALSE);
END;
```

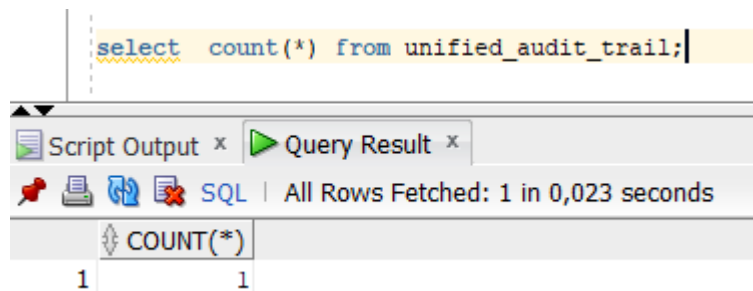
Функция очистит полностью весь унифицированный журнал, не учитывая *last archive timestamp* (как я понял, этот параметр связан с другим - *audit\_trail\_purge\_interval*, который указывает интервал в часах, как часто очищать журнал аудита. То есть если *use\_last\_arch\_timestamp* указан в TRUE, как 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:



COUNT(*)	
1	1224

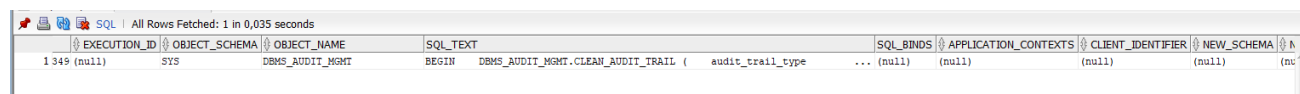
После:



```
select count(*) from unified_audit_trail;
```

COUNT(*)	
1	1

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



EXECUTION_ID	OBJECT_SCHEMA	OBJECT_NAME	SQL_TEXT	SQL_BINDS	APPLICATION_CONTEXTS	CLIENT_IDENTIFIER	NEW_SCHEMA
1349 (null)	SYS	DBMS_AUDIT_MGMT	BEGIN DBMS_AUDIT_MGMT.CLEAN_AUDIT_TRAIL ( audit_trail_type ... (null) (null) (null) (null) (null)	(null)	(null)	(null)	(null)

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

```
BEGIN
  DBMS_AUDIT_MGMT.CLEAN_AUDIT_TRAIL (
    audit_trail_type      => DBMS_AUDIT_MGMT.AUDIT_TRAIL_FGA_STD,
    use_last_arch_timestamp => FALSE);
END;
```

До:

```
SELECT count(*) total FROM dba_fga_audit_trail;
```

TOTAL	
1	21

После:

TOTAL	
1	0

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 :)