«Национальный исследовательский ядерный университет «МИФИ»

Институт интеллектуальных кибернетических систем

Кафедра №42 «Криптология и кибербезопасность»

Отчёт по лабораторной работе № 2\_1

"Безопасность базы данных"

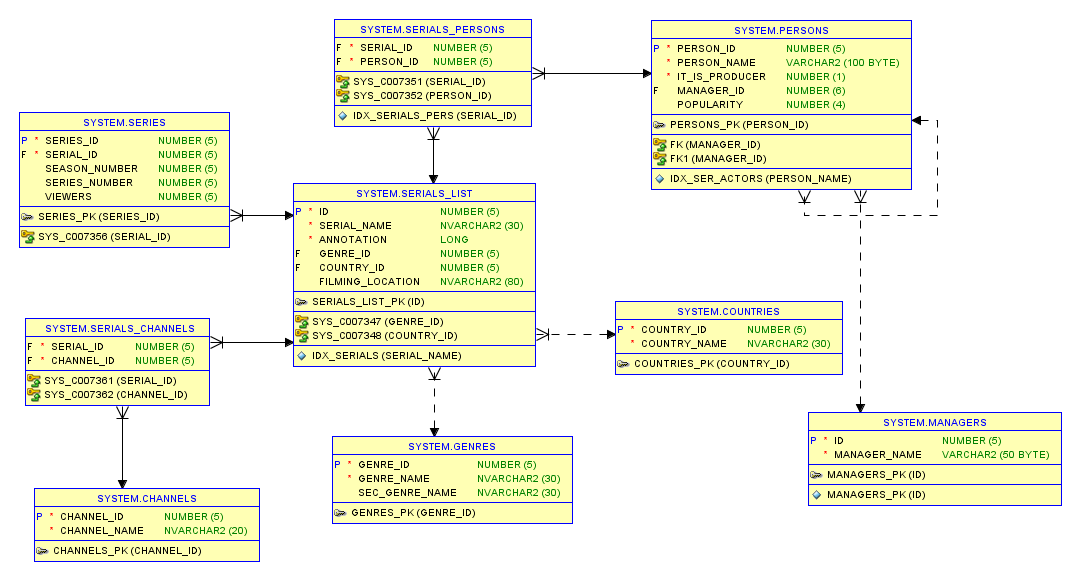
Студент: Нурматов С.Р.

Группа: Б17-565

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

Москва 2020

**Диаграмма сущностей**

****

**1. Создать дополнительного пользователя в разрабатываемой подключаемой базе данных (один пользователь уже должен существовать). Необходимо предоставить этому пользователю доступ к разработанным таблицам. Для этого допустимо предоставить новому пользователю привилегии SELECT ANY TABLE, INSERT ANY TABLE, UPDATE ANY TABLE, DELETE ANY TABLE. Можно также предоставить доступ к каждой таблице индивидуально;**

create user c##AzureUser identified by \*\*\*\*\*\*\*;

User C##AZUREUSER created.

create role c##new\_role;

GRANT SELECT ANY TABLE TO c##new\_role

GRANT UPDATE ANY TABLE TO c##new\_role

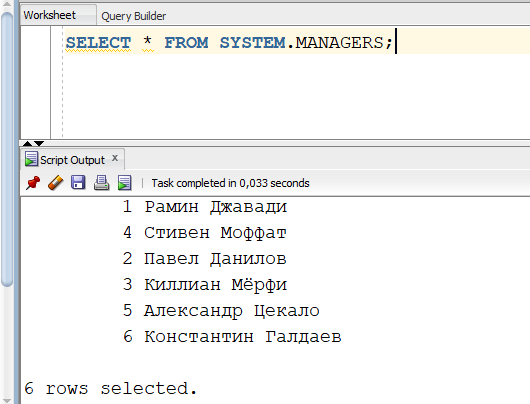
GRANT INSERT ANY TABLE TO c##new\_role

GRANT DELETE ANY TABLE TO c##new\_role

GRANT c##new\_role to c##azureuser;

В результате будет создана роль, которой будут предоставлены следующие привилегии: записи и обновление таблиц, выборка и удаление таблиц, а также создание сессии (пп. 3).

**2. Убедиться, что созданный пользователь имеет доступ к созданным ранее таблицам. Для этого попробовать выполнить инструкцию SELECT. Важно, что для доступа к таблицам других пользователей необходимо явно указать владельца таблицы: SELECT \* FROM testuser.testtable;**

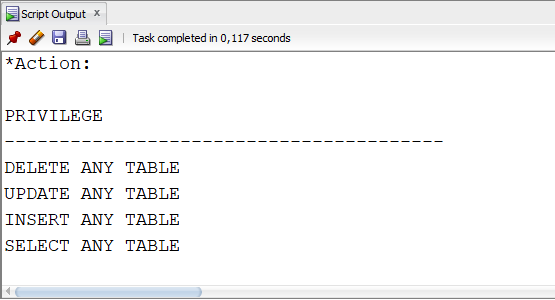
****

В результате будет выведена таблица с именами менеджеров от лица нового пользователя.

select privilege

from dba\_sys\_privs

where grantee= 'C##AZUREUSER'



**3. Подключиться к экземпляру базы данных от лица каждого из пользователей одновременно (потребуется два одновременно работающих процесса SQL\*Plus);**

grant create session to C##azureuser

connect c##azureuser/qwe8112

connect system/qwe8112

Было выполнено подключение к экземпляру базы данных от имени нового пользователя.

**4. Начать транзакцию от лица первого пользователя; для этого выполнить изменение данных в таблице;**

INSERT ALL

INTO managers(id, manager\_name)VALUES (1, 'Рамин Джавади')

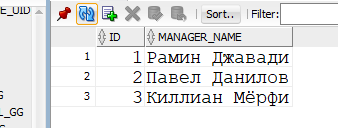
INTO managers(id, manager\_name)VALUES (2, 'Павел Данилов')

INTO managers(id, manager\_name)VALUES (3, 'Киллиан Мёрфи')

SELECT \* FROM DUAL;

COMMIT;

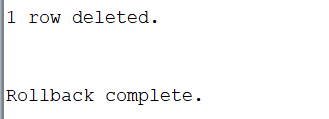
От лица 1 пользователя была начата транзакция добавления менеджеров в таблицу managers. С помощью команды COMMIT были сохранены текущие изменения.



DELETE FROM managers

where id = 1;

ROLLBACK;



Так же от лица первого пользователя была удалена одна строка с именем менеджера и выполнена команда ROLLBACK, которая отменяет выполненные изменения.

SAVEPOINT SP1;

DELETE FROM Managers WHERE ID=1;

SAVEPOINT SP2;

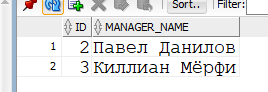
DELETE FROM Managers WHERE ID=2;



SAVEPOINT SP3;

DELETE FROM Managers WHERE ID=3;

ROLLBACK TO SP2;



Так как SP2 была создана после первого удаления, последняя два удаления будут отменены

Можно создавать точки, к которой группы транзакций могут откатиться с помощью команды SAVEPOINT SAVEPOINT\_NAME. То есть при создании той или иной точки можно выполнить операцию удаления и в дальнейшем с помощью команды ROLLBACK SAVEPOINT\_NAME вернуть отменить последние изменения. Если данная точка больше не нужна, можно освободить ее RELEASE SAVEPOINT SAVEPOINT\_NAME.

**5. Выполнить запрос к изменённым данным от имени обоих пользователей. Сравнить результаты и объяснить результаты сравнения;**

(От лица 1 пользователя)

SAVEPOINT SP3;

DELETE FROM Managers WHERE ID=3;

SELECT \* FROM system.managers (От лица 2 пользователя)

ID MANAGER\_NAME

---------- --------------------------------------------------

1 Рамин Джавади

2 Павел Данилов

3 Киллиан Мёрфи

SELECT \* FROM managers (От лица 1 пользователя)

ID MANAGER\_NAME

---------- --------------------------------------------------

1 Рамин Джавади

2 Павел Данилов

В результате в таблице были удалены данные от лица нового пользователя, но в таблице старого пользователя они не удалились.

**6. Выполнить откат транзакции первого пользователя с помощью инструкции ROLLBACK. Повторно выполнить п. 5;**

От лица 1 пользователя

ROLLBACK;

SELECT \* FROM system.channels (От лица 2 пользователя)

ID MANAGER\_NAME

---------- --------------------------------------------------

1 Рамин Джавади

2 Павел Данилов

3 Киллиан Мёрфи

SELECT \* FROM channels (От лица 1 пользователя)

ID MANAGER\_NAME

---------- --------------------------------------------------

1 Рамин Джавади

2 Павел Данилов

3 Киллиан Мёрфи

RPLLBACK используется для отмены транзакций, которые не были сохранены в базе данных. Команда используется только для отмены транзакций с момента выполнения последней команды COMMIT или ROLLBACK.

В результате данные, которые были до транзакции восстанавливаются (От лица первого пользователя!)

**7. От лица первого пользователя выполнить ещё одно изменение данных в таблицах. Выполнить фиксацию транзакции при помощи инструкции COMMIT. Повторно выполнить п. 5;**

От лица первого пользователя

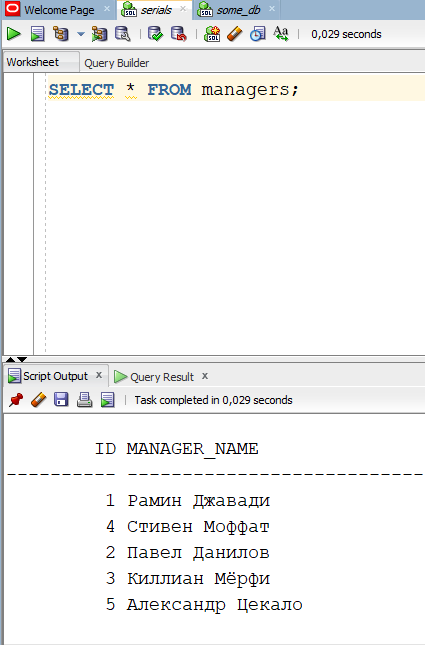
insert all

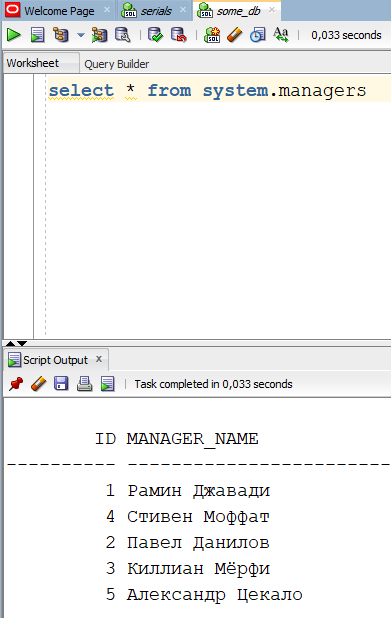
into managers(id, manager\_name) values (4, 'Стивен Моффат')

into managers(id, manager\_name) values (5, 'Александр Цекало')

SELECT \* FROM dual;

COMMIT





В результате данные запрошенные от лица 2 пользователя изменились, так как транзакция была сохранена в базе данных (От лица 1 пользователя!).

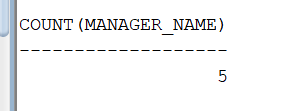
**8. Если от лица второго пользователя начата транзакция, завершить её с помощью инструкции ROLLBACK или COMMIT. Начать от лица данного пользователя новую сериализованную транзакцию, используя инструкцию SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;**

От лица 2 пользователя 1 уровень изоляции транзакции установлен на SERIALIZABLE

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE

select count(manager\_name)

from system.managers



От лица первого пользователя добавлен еще один менеджер.

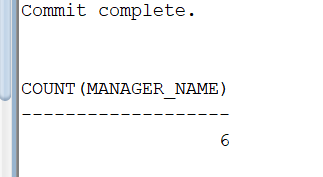
insert all

into managers(id, manager\_name) values(6, 'Константин Галдаев')

select \* from dual;

commit;

select count(manager\_name) from managers;

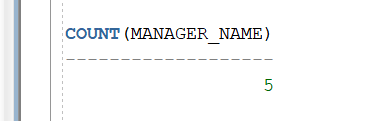


Второй сеанс не может пока посчитать зафиксированное количество менеджеров

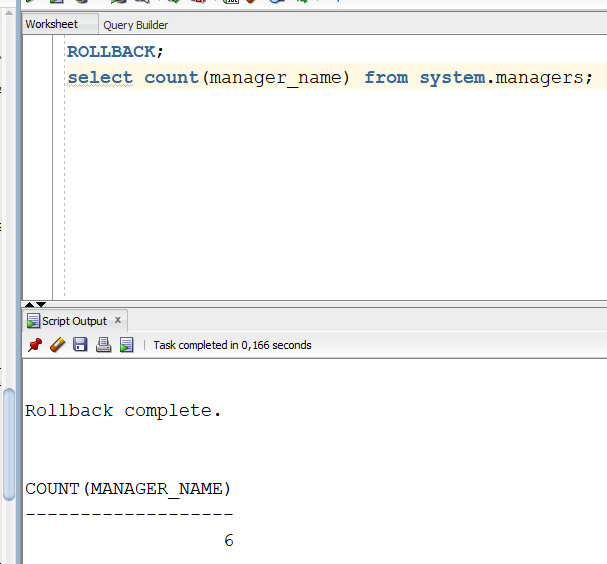
От лица 2 пользователя

set transaction isolation level SERIALIZABLE;

select count(manager\_name) from system.managers;



Несмотря на то, что данные были изменены и зафиксированы от лица 1 пользователя, они не появились в данных старого пользователя, что доказывает работоспособность set transaction isolation level SERIALIZABLE;



Транзакция завершается с помощью команды ROLLBACK. Теперь данные можно считать фиксированными.

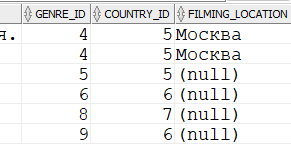
**9. Повторить пп. 7 и 5**

От лица 1 пользователя

UPDATE serials\_list SET Filming\_location = 'Москва' where id = 1

UPDATE serials\_list SET Filming\_location = 'Москва' where id = 2

COMMIT



От лица 2 пользователя

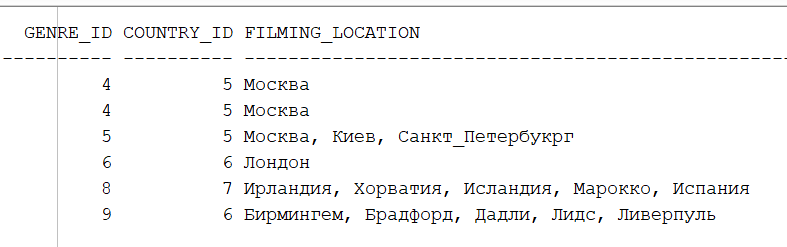
UPDATE system.serials\_list

SET Filming\_location = 'Москва, Киев, Санкт\_Петербукрг' where id = 3

…

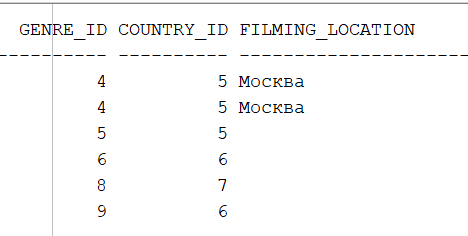
…

SELECT \* FROM SYSTEM.serials\_list



От лица 1 пользователя

SELECT \* FROM serials\_list

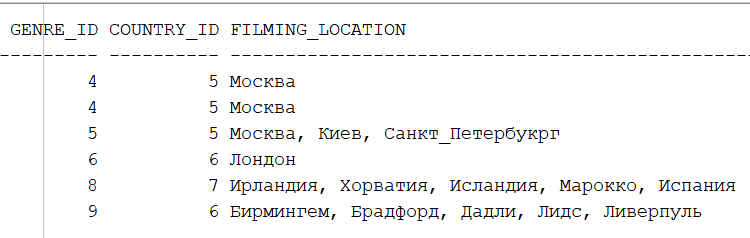


От лица 2 пользователя

COMMIT

От лица 1 пользователя

SELECT \* FROM serials\_list



Как мы видим, в результате изменения, которые были выполнены от лица 2 пользователя не видимы от лица старого пользователя, так как транзакция не была выполнена. Только после сохранения данных в базе данных от лица 2 пользователя данные будут обновлены в таблице 1 пользователя.

**Залючение**

В ходе выполнения данной работы:

Были созданы новые пользователи в данной базе данных, которым были предоставлены соответствующие привилегии.

Были выполнены транзакции от лица 1 и 2 пользователей.

Были использованы команды ROLLBACK, COMMIT для сравнения результатов

Был показан принцип работы инструкции SET TRANSACTION ISOLATION LEVEL SERIALIZABLE

Сcылка на GitHub:

https://github.com/Nurmatov-07/DataBase2020/tree/master