

Методические указания
по лабораторной работе №3
на тему: «Защита базы данных»

по дисциплине: Информационная безопасность баз данных

Санкт-Петербург
2024

Цель работы

Получение навыков созданию примитивных систем мониторинга, разграничения доступа и шифрования средствами СУБД.

Задание

1. Задачи по мониторингу БД:

- Создайте таблицу-лог, отдельную от ваших основных сущностей БД.
- Создайте для каждой основной таблицы в вашей БД триггер, который срабатывает при любых изменениях в БД (вставка новых данных, изменение существующих записей, удаление кортежей из таблицы). При срабатывании триггер должен вносить в таблицу-лог информацию о том, когда было произведено изменение, со стороны какой роли поступил запрос, какие кортежи поменялись, старые и новые значения.
- Прдемонстрируйте работу системы логирования для различных операций и отношений.

2. Задачи по шифрованию данных.

- Создайте таблицу с секретными данными, отдельно от ваших основных сущностей. Например, это может быть таблица с токенами или ключами доступа, для каждого класса-пользователей.
- Зашифруйте содержимое данной таблицы, в качестве алгоритма шифрования используйте любой симметричный алгоритм шифрования. Ключ шифрования для данной таблицы не должен храниться в ИС. Ключ шифрования может быть получен из индивидуального пароля для дешифрования суперпользователя (пароль не связан с паролем для входа в СУБД). Индивидуальный пароль суперпользователя и ключ шифрования может быть связан через одностороннюю функцию. Например, пусть индивидуальный пароль комбинация «!stroNgpsw31234», считаем от данного пароля детерминированную хэш-функцию (например, sha-256), полученный хэш-используем как ключ шифрования/дешифрования для симметричного алгоритма шифрования таблицы с секретными данными (например, для AES-256)
- Демонстрируем, что даже обладая полными правами администратора, но без знания индивидуального пароля невозможно получить содержимое таблицы с секретными данными

3. Задачи по разграничению доступа в БД:

- Создайте в СУБД как минимум 2 роли (суперпользователь не считается) для каждого из классов потребителей информации;
- С помощью внутренних инструментов СУБД для каждой роли определите набор привилегий по отношению к таблицам вашей БД. Руководствуйтесь принципом минимальных привилегий, если определенному классу потребителей не нужен доступ к определенным таблицам/атрибутам (список

задач БД, составленный в рамках 1 ЛР), то доступ к этим таблицам/атрибутам не предоставляется. Разграничиваем доступ к представлениям, созданным в 1 ЛР, а также таблицам логирования (таблицы логирования может просматривать только суперпользователь)

- Продемонстрируйте работу вашей системы разграничения доступа. Зайдите за каждую из ролей и покажите доступные со стороны каждой роли отношения.

Пример отчета по лабораторной работе №3.

Задачи по мониторингу операций в БД:

Создадим триггерную функцию *logging()*, которая будет срабатывать при изменениях данных и сохранять тип операции, время, пользователя и выполненные изменения.

```
CREATE OR REPLACE FUNCTION logging() RETURNS TRIGGER AS $logging$
BEGIN
  IF (TG_OP = 'DELETE') THEN
    INSERT INTO public.main_log (operation_type, operation_date, user_operator, changed_data)
    VALUES ('D', now(), current_user, row_to_json(OLD));
  ELSIF (TG_OP = 'UPDATE') THEN
    INSERT INTO public.main_log (operation_type, operation_date, user_operator, changed_data)
    VALUES ('U', now(), current_user, row_to_json(NEW));
  ELSIF (TG_OP = 'INSERT') THEN
    INSERT INTO public.main_log (operation_type, operation_date, user_operator, changed_data)
    VALUES ('I', now(), current_user, row_to_json(NEW));
  END IF;
  RETURN NULL;
END;
$logging$ LANGUAGE plpgsql;
```

Создадим таблицу-лог «main_log», состоящую из следующих атрибутов:

```
CREATE TABLE public.main_log
(
  log_item_id integer NOT NULL,
  operation_type text,
  operation_date text,
  user_operator text,
  changed_data text,
  PRIMARY KEY (log_item_id)
);
```

Сделаем атрибут «log_item_id» автогенерируемым идентификатором кортежа в таблице:

```
ALTER TABLE IF EXISTS public.main_log
```

```
ALTER COLUMN log_item_id ADD GENERATED BY DEFAULT AS IDENTITY;
```

Создаем для каждой таблицы («classes», «groups», «specialty», «subject», «teachers») в БД триггер, который вызывает триггерную функцию *logging()*, по следующему шаблону:

```
CREATE TRIGGER logging_exec
AFTER INSERT OR UPDATE OR DELETE ON classes
FOR EACH ROW EXECUTE FUNCTION logging();
```

Продemonстрируем работу созданной системы логирования для случайных таблиц. Например, вставим данные таблицы *classes* и выведем содержимое лога «main_log».

Вставка данных:

```
INSERT INTO public.classes (classes_id, classes_start_time, classes_end_time, classroom,
building, subject_id)
VALUES ('10'::integer, '8'::text, '9'::text, '232'::text, '1231'::text, '1'::integer);
```

Таблица-лог при вставке данных, представлена на Рисунке 1:

	log_item_id [PK] integer	operation_type text	operation_date text	user_operator text	changed_data text
1	1	I	2024-09-02 06:22:11.909381+03	postgres	{'classes_id':10,'classes_start_time':'8.00','classes_end_time':'9.00','classroom':'404','building':'Kronverskii, 16','subject_id':6}

Рисунок 1 – Логирование операции вставки

Изменим данные в таблице «Преподаватели»

```
UPDATE public.teachers SET
email = 'vvv_new_mail@mail.ru'::text WHERE
teacher_id = 3277122;
```

Удалим одну из записей из таблицы «Занятия»

```
DELETE FROM public.classes
WHERE classes_id IN (2);
```

Снова проверим работоспособность таблицы-лога (Результат на Рисунке 2.):

	log_item_id [PK] integer	operation_type text	operation_date text	user_operator text	changed_data
1	1	I	2024-09-02 06:22:11.909381+03	postgres	{'classes_id':10,'classes_start_time':'8.00','classes_end_time':'9.00','classroom':'404','building':'Kronverskii, 16','subject_id':6}
2	2	U	2024-09-02 06:30:12.825738+03	postgres	{'teacher_id':3277122,'teacher_full_name':'Vasykov Vladimir Vasilievich','department':'TSG','email':'vvv_new_mail@mail.ru','phone':'(922)34783823','consultation_time':'8-14, friday','working'}
3	3	D	2024-09-02 06:34:10.673359+03	postgres	{'classes_id':2,'classes_start_time':'11.00','classes_end_time':'11.30','classroom':'304','building':'Pesochnaya, 14','subject_id':1}

Рисунок 2 – Логирование всех операций

2. Задачи по шифрованию данных.

Создадим таблицу с секретными данными. Например, это может быть таблица с токенами или ключами доступа, для каждого пользователей.

```
CREATE TABLE public.secret_data
```

```
(
```

```
"ID" integer NOT NULL GENERATED BY DEFAULT AS IDENTITY,
```

```

username text,
secret_token text,
PRIMARY KEY ("ID")
);

```

Загружаем модуль «pgcrypto», если он не установлен. Заполним таблицу примерами зашифрованных данных. Пусть индивидуальным паролем админа будет «personal_admin_pass», посчитаем hash sha-256 от данного пароля, получим «9f86d081884c7d659a2feaa0c55ad015a3bf4f1b2b0b822cd15d6c15b0f00a08». Полученный хэш будем использовать как симметричный ключ шифрования. Вставим данные в зашифрованном виде:

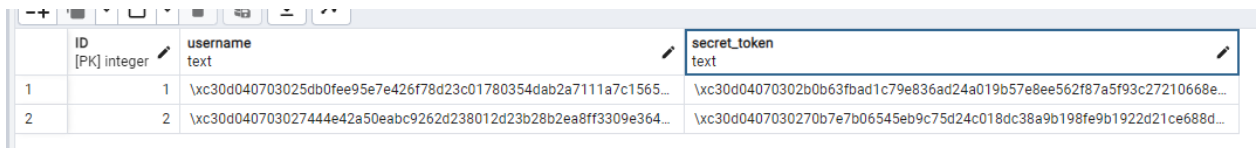
```

INSERT INTO secret_data (username, secret_token)
VALUES
(pgp_sym_encrypt('gena_kozlov',
'9f86d081884c7d659a2feaa0c55ad015a3bf4f1b2b0b822cd15d6c15b0f00a08'),
pgp_sym_encrypt('token_4fsdf43fdaf43r2fdsa',
'9f86d081884c7d659a2feaa0c55ad015a3bf4f1b2b0b822cd15d6c15b0f00a08')),

(pgp_sym_encrypt('vasykov',
'9f86d081884c7d659a2feaa0c55ad015a3bf4f1b2b0b822cd15d6c15b0f00a08'),
pgp_sym_encrypt('token_13gdfss4fa232fdsadfsa',
'9f86d081884c7d659a2feaa0c55ad015a3bf4f1b2b0b822cd15d6c15b0f00a08'));

```

Зашифрованные значения таблицы «secret_data» в итоге имеют вид, представленный на Рисунке 3.



ID	username	secret_token
1	\xc30d040703025db0fee95e7e426f78d23c01780354dab2a7111a7c1565...	\xc30d04070302b0b63fbad1c79e836ad24a019b57e8ee562f87a5f93c27210668e...
2	\xc30d040703027444e42a50eabc9262d238012d23b28b2ea8ff3309e364...	\xc30d0407030270b7e7b06545eb9c75d24c018dc38a9b198fe9b1922d21ce688d...

Рисунок 3 – Таблица «secret_data»

3. Задачи по разграничению доступа в БД.

Согласно заданию, необходимо создать как минимум 2 групповые роли для каждого из классов потребителей информации. В первой лабораторной были выделены потребители «Студент» и «Персонал». Также создадим групповую роль «Админ», необходима для 4 ЛР, «Админ» должен иметь возможность просматривать все таблицы и представления в БД.

Код создания групповых ролей:

```

CREATE ROLE student_group_role WITH
    NOLOGIN
    NOSUPERUSER
    NOCREATEDB
    NOCREATEROLE

```

INHERIT

Согласно целям проектирования БД и исходя из принципа представления наименьших привилегий, выделим групповой роли «Студенты» привилегии на просмотр только представлений «compact_shedule» и «consultations»:

```
GRANT SELECT ON TABLE public.compact_shedule TO student_group_role;  
GRANT SELECT ON TABLE public.consultations TO student_group_role;
```

Аналогично создадим групповую роль для персонала (*staff_group_role*) и предоставим доступ для данной групповой роли только к представлениям «*speciality_and_contacts*» и «*speciality_and_teachers*»:

```
CREATE ROLE staff_group_role WITH  
    NOLOGIN  
    NOSUPERUSER  
    NOCREATEDB  
    NOCREATEROLE  
    INHERIT
```

```
GRANT SELECT ON TABLE public.speciality_and_contacts TO staff_group_role;  
GRANT SELECT ON TABLE public.speciality_and_teachers TO staff_group_role;
```

И наконец, создадим и определим привилегии для групповой роли «Админ»:

```
CREATE ROLE admin_group_role WITH  
    NOLOGIN  
    NOSUPERUSER  
    NOCREATEDB  
    NOCREATEROLE  
    INHERIT
```

```
GRANT SELECT ON TABLE  
    classes,  
    "groups",  
    main_log,  
    secret_data,  
    specialty,  
    subject,  
    teachers,  
    speciality_and_contacts,  
    speciality_and_teachers,  
    compact_shedule,  
    consultations  
TO admin_group_role;
```

Теперь создадим по одному пользователю для каждой из групповых ролей. Выделим им привилегии группы и продемонстрируем работу предлагаемой системы разграничения доступа на примере доступа к случайным таблицам.

Начнем с пользователя-студента, который будет обладать привилегиями групповой роли «*student_group_role*». Создадим индивидуальную роль «*Romanov_student*» для конкретного студента с возможностью LOGIN:

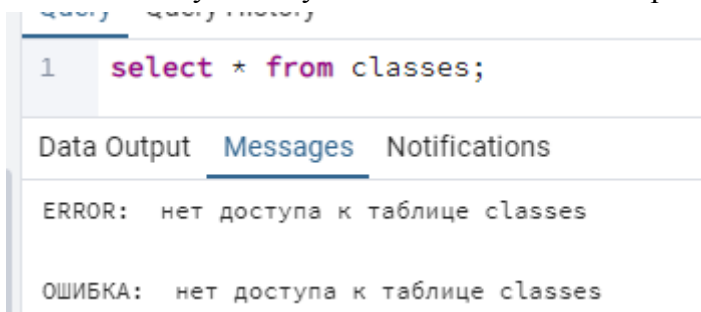
```
CREATE ROLE "Romanov_student" WITH  
    LOGIN  
    NOSUPERUSER  
    NOCREATEDB  
    NOCREATEROLE
```

```
GRANT student_group_role TO "Romanov_student";
```

Зайдет со стороны индивидуальной роли "Romanov_student" и проверим доступность содержимого таблиц.

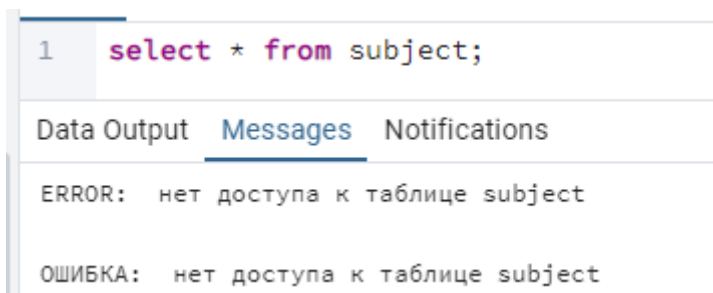
```
SET ROLE "Romanov_student";
```

Попытка доступа к случайным таблицам со стороны "Romanov_student":



```
1 select * from classes;
```

Data Output	Messages	Notifications
ERROR: нет доступа к таблице classes		
ОШИБКА: нет доступа к таблице classes		



```
1 select * from subject;
```

Data Output	Messages	Notifications
ERROR: нет доступа к таблице subject		
ОШИБКА: нет доступа к таблице subject		

Query Query History

```
1 select * from speciality_and_contacts;
```

Data Output Messages Notifications

ERROR: нет доступа к представлению speciality_and_contacts

ОШИБКА: нет доступа к представлению speciality_and_contacts

Query Query History

```
1 select * from compact_shedule;
```

Data Output Messages Notifications

	subject_name text	subject_type text	classes_start_time text	classes_end_time text	building text	teacher_full_name text
1	Database Security	lection	8.20	10.00	Pesochnaya, ...	Kozlov Genadii Andreevich
2	Database Security	labs	11.40	13.10	Kronverskii, 16	Dementiev Sergey Danilovich
3	Database Security	labs	15.00	16.30	Birzhevaya, 6	Dementiev Sergey Danilovich
4	Database Security	lection	18.00	19.00	Kronverskii, 16	Dementiev Sergey Danilovich

Доступны только представления «compact_shedule» и «consultations».

Аналогично, создадим и проверим индивидуальную роль «kozlov_teacher» для групповой роли «staff_group_role»:

```
CREATE ROLE kozlov_teacher WITH
    LOGIN
    NOSUPERUSER
    NOCREATEDB
```

```
grant staff_group_role to kozlov_teacher;
```

```
set role kozlov_teacher;
```

Попытки доступа:

Query Query History

```
1 select * from teachers;
```

Data Output Messages Notifications

ERROR: нет доступа к таблице teachers

ОШИБКА: нет доступа к таблице teachers

SQL state: 42501


```
1 select * from main_log;
```

Data Output Messages Notifications

ERROR: нет доступа к таблице main_log

ОШИБКА: нет доступа к таблице main_log
SQL state: 42501

```
1 select * from consultations;
```

Data Output Messages Notifications

ERROR: нет доступа к представлению consultations

ОШИБКА: нет доступа к представлению consultations

```
1 select * from speciality_and_contacts;
```

Data Output Messages Notifications

	specialty_name text	group_id integer	course_number integer	admin_full_name text	admin_contacts text
1	Information Security	3351	3	Vasiliev Andrei	phone 921-323-23-44
2	Information Security	2311	2	Krivov Dmitrii	mail kdmitr@gmail.com
3	IoT Security	1131	1	Ivanov Ivan	ivan@mail.ru

Для индивидуальной роли «Kozlov_teacher» доступны только представления «speciality_and_contacts» и «speciality_and_teachers».

Создадим, индивидуальную роль «petya_admin».

```
CREATE ROLE petya_admin WITH  
    LOGIN  
    NOSUPERUSER  
    NOCREATEDB  
    NOCREATEROLE
```

```
grant admin_group_role TO petya_admin;  
set role petya_admin;
```

Попытки доступа:

1select * from main_log;

Data Output

Messages

Notifications

≡+

▼

▼

SQL

	log_item_id [PK] integer	operation_type text	operation_date text	user_operator text	changed_data text
1	1	I	2024-09-02 06:22:11.909381+03	postgres	{"classes_id":10,"classes_start_time":"8.00","classes_en
2	2	U	2024-09-02 06:30:12.825738+03	postgres	{"teacher_id":3277122,"teacher_full_name":"Vasykov VI
3	3	D	2024-09-02 06:34:10.673359+03	postgres	{"classes_id":2,"classes_start_time":"11.00","classes_en

1

select * from secret_data;

Data Output

Messages

Notifications

≡+

📄

▼

📋

▼

🗑️

🔍

⬇️

📶

SQL

	ID [PK] integer	username text
1	1	\xc30d040703025db0fee95e7e426f78d23c01780354dab2a7111a7c1565a0a6f760a730f3b1b8d59be26d6t
2	2	\xc30d040703027444e42a50eabc9262d238012d23b28b2ea8ff3309e364fb28dffe2212318eb33a03a4f0b6

1

select * from consultations;

Data Output

Messages

Notifications

≡+

📄

▼

📋

▼

🗑️

🔍

⬇️

📶

SQL

	subject_name text	teacher_full_name text	department text	email text	phone text	consultation_time text	working_bulding text
1	Database Security	Kozlov Genadii Andreevich	PBKS	kga@gmail.com	(901)3235511	12-15, every monday	Kronverskii, 14, classroom 404
2	Database Security	Dementiev Sergey Danilovich	KTU	dsd@gmail.com	(911)8463244	18-20, every monday	Birzhevaya, 16, classroom 412
3	Cryptography	Dementiev Sergey Danilovich	KTU	dsd@gmail.com	(911)8463244	18-20, every monday	Birzhevaya, 16, classroom 412
4	Network Security	Vasykov Vladimir Vasilievich	TSG	vvv_new_mail@mail.ru	(922)34783823	8-14, friday	Pesochnaya, 14, class 300
5	Application Security	Vasykov Vladimir Vasilievich	TSG	vvv_new_mail@mail.ru	(922)34783823	8-14, friday	Pesochnaya, 14, class 300
6	Coding Theory	Vasykov Vladimir Vasilievich	TSG	vvv_new_mail@mail.ru	(922)34783823	8-14, friday	Pesochnaya, 14, class 300

1select * from speciality_and_teachers;

Data Output

Messages

Notifications

≡+

📄

▼

📋

▼

🗑️

🔍

⬇️

📶

SQL

	specialty_name text	department text	group_id integer	subject_name text	teacher_full_name text
1	IoT Security	TSG	1131	Database Security	Kozlov Genadii Andreevich
2	IoT Security	TSG	1131	Database Security	Dementiev Sergey Danilovich
3	Information Security	KTU	2311	Cryptography	Dementiev Sergey Danilovich
4	Information Security	KTU	2311	Network Security	Vasykov Vladimir Vasilievich
5	Information Security	KTU	3351	Application Security	Vasykov Vladimir Vasilievich
6	Information Security	KTU	3351	Coding Theory	Vasykov Vladimir Vasilievich