



ИНСТИТУТ ИНТЕЛЛЕКТУАЛЬНЫХ КИБЕРНЕТИЧЕСКИХ СИСТЕМ

**Кафедра
«Криптология и кибербезопасность»**

Лабораторная работа №1

по предмету «Безопасность систем баз данных»

Выполнил студент группы Б20-505

Сорочан Илья

Москва – 2024

Содержание

1. Диаграмма сущностей и разделение на схемы	3
2. Роли	5
3. Проверка ролей	6
3.1. Краткий экскурс в postgres	6
3.2. Просмотр pg_roles	6
3.3. pdb_role	7
3.4. sales_role	7
3.5. staff_role	8
3.6. logistics	8
Заключение	10
Приложение А	11
Приложение В	14

1. Диаграмма сущностей и разделение на схемы

Для начала напомним, что база данных, разработанная в предыдущем семестре используется в гипотетическом канцелярском магазине. Product это конкретная единица товара, Product Type - «род» товара. Так же учитываются поставки и продажи, на оба события назначается ответственный работник/кассир.

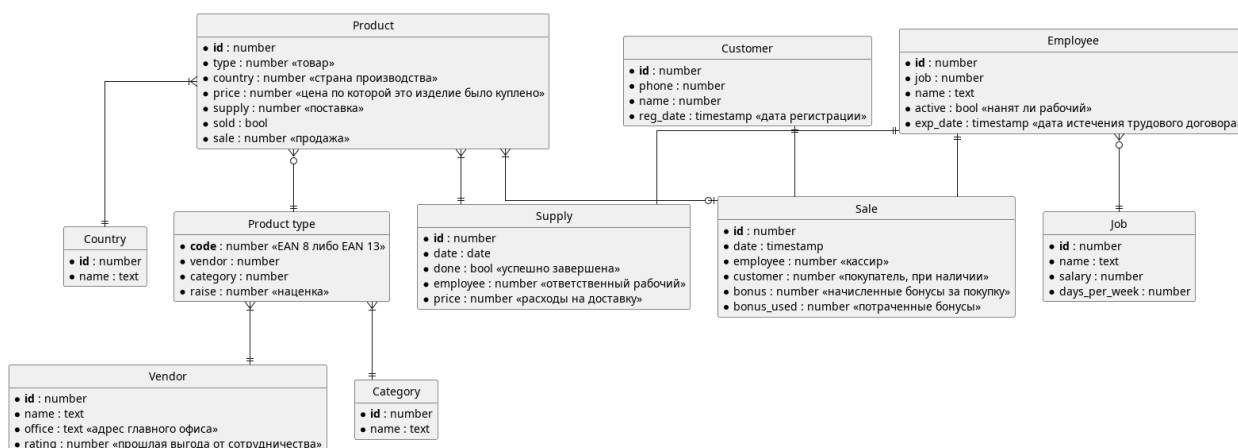


Рис. 1. Диаграмма сущностей

Далее по заданию необходимо разделить сущности на схемы при необходимости. Я посчитал полезным ограничить примерные границы для ролей следующим образом:

- PDB (Product DataBase) - все что относится к данным о продуктах в целом:
 - Product Type
 - Country
 - Vendor
 - Category
- inventory - продукты на витринах:
 - Product
- sales - все что относится к продажам:
 - Sale
 - Customer
- staff - менеджмент сотрудников:
 - Employee

- Job
- logistics - поставки:
- Supply

Можно заметить, что в inventory и logistics только по одному элементу. Это сделано для того что бы все оставалось организованным и при этом не хранилось в public, так как с полными привилегиями по умолчанию можно легко накосячить.

Для того что бы добавить схему необходимо прописать

```
CREATE SCHEMA pdb;
```

И после использовать префикс в виде схемы в названии таблицы (например вместо product pdb.product).

2. Роли

После разбиения на схемы было решено ввести 4 роли: роль для модификации pdb, роль для продаж (кассиры), роль для hr'ов и роль для управления логистикой (какие-нибудь менеджеры):

- pdb_role
 - SELECT, INSERT, UPDATE, DELETE на pdb
- sales_role - роль «продавца», списывает продукты, добавляет sale'ы.
 - SELECT, INSERT, UPDATE, DELETE на sales
 - SELECT, UPDATE на inventory (там только bool поставить надо)
- staff_role - роль для HR'ов:
 - SELECT, INSERT, UPDATE, DELETE на staff
 - SELECT на sales (для того что бы потенциально смотреть насколько каждый сотрудник эффективен)
- logistics_role - роль для управления поставками:
 - SELECT, INSERT, UPDATE, DELETE на logistics и inventory

Важно отметить, что особо глубокого смысла в данном разделении нет, я просто подумал про справедливые, но строгие требования и написал их как вижу.

Также замечу, что ни одна роль не имеет системных привелегий. Это потому, что они здесь не должны быть. Данная архитектура уже предусматривает добавление товара и его параметров. Если все-таки бд необходимо будет перестроить, то это уже будут кардинальные изменения. В таком случае разумно использовать суперпользователя.

3. Проверка ролей

3.1. Краткий экскурс в postgres

При выполнении лабораторных я использовал docker образ postgres. Что бы поднять его у себя сначала пуллим, а затем, собственно, поднимаем (user:user простой тестовый пароль):

```
sudo docker pull postgres
sudo docker run -itd -e POSTGRES_USER=user -e POSTGRES_PASSWORD=user
-p 5432:5432 -v /data:/var/lib/postgresql/data --name postgresql
postgres
```

После этого подключится к серверу в качестве user с той же машины можно будет следующей командой:

```
PGPASSWORD=user psql -U user -d mydb -h localhost
```

Если у вас пишет, что mydb не существует, то просто создайте ее от имени user:

```
PGPASSWORD=user psql -U user -h localhost -c "CREATE DATABASE mydb;"
```

Напоследок отмечу, что выполнить просто команды можно через -с, выполнить файл через -f. Файл setup.sql используется для создания тестовой бд.

3.2. Просмотр pg_roles

Для того что бы быстро задавать использовался файл setup.sql (см приложение).

Зайдем на сервер и посмотрим pg_roles

```
~/Documents/bsbd_labs_s2/playground master ?3 > PGPASSWORD=user psql -U user -d mydb -h localhost -c "SELECT rolname, rolsuper, rolinherit, rolcreatorole, rolcreatedb, rolcanlogin
FROM pg_catalog.pg_roles;"
```

rolname	rolsuper	rolinherit	rolcreatorole	rolcreatedb	rolcanlogin
pg_database_owner	f	t	f	f	f
pg_read_all_data	f	t	f	f	f
pg_write_all_data	f	t	f	f	f
pg_monitor	f	t	f	f	f
pg_read_all_settings	f	t	f	f	f
pg_read_all_stats	f	t	f	f	f
pg_stat_scan_tables	f	t	f	f	f
pg_read_server_files	f	t	f	f	f
pg_write_server_files	f	t	f	f	f
pg_execute_server_program	f	t	f	f	f
pg_signal_backend	f	t	f	f	f
pg_checkpoint	f	t	f	f	f
pg_use_reserved_connections	f	t	f	f	f
pg_create_subscription	f	t	f	f	f
user	t	t	t	t	t
pdb_role	f	t	f	f	t
sales_role	f	t	f	f	t
staff_role	f	t	f	f	t
logistics_role	f	t	f	f	t

(19 rows)

Рис. 2. pg_roles

Как можно убедиться, ни одна роль не имеет системных привелегий. Теперь пройдемся по ролям.

3.3. pdb_role

```
-- should allow
SELECT vendor FROM pdb.product_type;

-- should deny
SELECT salary, days_per_week FROM sales.employee;
```

```
~/Documents/bsbd_labs_s2/playground master ?3 > psql -W -U pdb_role -d mydb -h localhost -f role_check/pdb.sql
Password:
vendor
-----
1
2
1
(3 rows)

psql:role_check/pdb.sql:5: ERROR: permission denied for schema sales
LINE 1: SELECT salary, days_per_week FROM sales.employee;
      ^
```

Рис. 3. Тест роли (1)

3.4. sales_role

```
-- should allow
SELECT price FROM inventory.product;

-- should deny
DELETE FROM inventory.product WHERE price > 1000;
```

```

~/Documents/bsbd_labs_s2/playground master ?3 > psql -W -U sales_role -d mydb -h localhost -f role_check/sales.sql
Password:
price
-----
 999
 899
1499
1299
 129
(5 rows)

psql:role_check/sales.sql:5: ERROR:  permission denied for table product

```

Рис. 4. Тест роли (2)

3.5. staff_role

```

-- should allow
SELECT name FROM staff.employee;

-- should deny
DELETE FROM inventory.product WHERE price > 1000;

```

```

~/Documents/bsbd_labs_s2/playground master ?3 > psql -W -U staff_role -d mydb -h localhost -f role_check/staff.sql
Password:
name
-----
Джон Стоун
Джек Воробей
Партос
(3 rows)

psql:role_check/staff.sql:5: ERROR:  permission denied for schema inventory
LINE 1: DELETE FROM inventory.product WHERE price > 1000;
                        ^

```

Рис. 5. Тест роли (3)

3.6. logistics

```

-- should allow
SELECT price FROM inventory.product;

-- should deny
SELECT salary, days_per_week FROM sales.employee;

```



```
~/Documents/bsbd_labs_s2/playground master ?3 > psql -W -U logistics_role -d mydb -h localhost -f role_check/logistics.sql
Password:
price
-----
 999
 899
1499
1299
 129
(5 rows)

psql:role_check/logistics.sql:5: ERROR:  permission denied for schema sales
LINE 1: SELECT salary, days_per_week FROM sales.employee;
        ^
```

Рис. 6. Тест роли (4)

Заключение

В данной лабораторной работе были рассмотрены такие важные аспекты PostgreSQL как схемы и роли. Сначала были выбраны наиболее подходящие схемы, а затем определены роли, соответствующие функционалу канцелярского магазина. Также было произведено подключение и проверка привилегий ролей, чтобы убедиться, что они соответствуют ожиданиям.

Приложение А

Код setup.sql

```
-- Tables

CREATE SCHEMA pdb;
CREATE SCHEMA staff;
CREATE SCHEMA sales;
CREATE SCHEMA logistics;
CREATE SCHEMA inventory;

CREATE TABLE pdb.vendor (
    id INTEGER PRIMARY KEY,
    name TEXT,
    office TEXT,
    rating INTEGER
);

CREATE TABLE pdb.category (
    id INTEGER PRIMARY KEY,
    name TEXT
);

CREATE TABLE pdb.product_type (
    code INTEGER PRIMARY KEY,
    vendor INTEGER,
    category INTEGER,
    raise INTEGER,
    FOREIGN KEY(vendor) REFERENCES pdb.vendor(id),
    FOREIGN KEY(category) REFERENCES pdb.category(id)
);

CREATE TABLE pdb.country (
    id INTEGER PRIMARY KEY,
    name TEXT
```

```
);
```

```
CREATE TABLE staff.job (  
    id INTEGER PRIMARY KEY,  
    name TEXT,  
    salary INTEGER,  
    days_per_week INTEGER  
);
```

```
CREATE TABLE staff.employee (  
    id INTEGER PRIMARY KEY,  
    job INTEGER,  
    name TEXT,  
    active BOOLEAN,  
    exp_date TIMESTAMP,  
    FOREIGN KEY(job) REFERENCES staff.job(id)  
);
```

```
CREATE TABLE sales.customer (  
    id INTEGER PRIMARY KEY,  
    phone INTEGER,  
    name TEXT,  
    reg_date TIMESTAMP  
);
```

```
CREATE TABLE sales.sale (  
    id INTEGER PRIMARY KEY,  
    date TIMESTAMP,  
    employee INTEGER,  
    customer INTEGER,  
    bonus INTEGER,  
    bonus_used INTEGER,  
    FOREIGN KEY(employee) REFERENCES staff.employee(id),  
    FOREIGN KEY(customer) REFERENCES sales.customer(id)  
);
```

```
CREATE TABLE logistics.supply (  

```

```

    id INTEGER PRIMARY KEY,
    date DATE,
    done BOOLEAN,
    employee INTEGER,
    price INTEGER,
    FOREIGN KEY(employee) REFERENCES staff.employee(id)
);

CREATE TABLE inventory.product (
    id INTEGER PRIMARY KEY,
    type INTEGER,
    country INTEGER,
    price INTEGER,
    supply INTEGER,
    sold BOOLEAN,
    sale INTEGER,
    FOREIGN KEY(type) REFERENCES pdb.product_type(code),
    FOREIGN KEY(country) REFERENCES pdb.country(id),
    FOREIGN KEY(supply) REFERENCES logistics.supply(id),
    FOREIGN KEY(sale) REFERENCES sales.sale(id)
);

-- Roles
-- NOTE: In production use ECRYPTED password set

CREATE ROLE pdb_role LOGIN PASSWORD '123';
GRANT SELECT, INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA pdb TO
pdb_role;
GRANT USAGE ON SCHEMA pdb TO pdb_role;

CREATE ROLE sales_role LOGIN PASSWORD '123';
GRANT SELECT, INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA sales TO
sales_role;
GRANT SELECT, UPDATE ON ALL TABLES IN SCHEMA inventory TO sales_role;
GRANT USAGE ON SCHEMA sales TO sales_role;
GRANT USAGE ON SCHEMA inventory TO sales_role;

```

```

CREATE ROLE staff_role LOGIN PASSWORD '123';
GRANT SELECT, INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA staff TO
staff_role;
GRANT SELECT ON ALL TABLES IN SCHEMA sales TO staff_role;
GRANT USAGE ON SCHEMA staff TO staff_role;
GRANT USAGE ON SCHEMA sales TO staff_role;

CREATE ROLE logistics_role LOGIN PASSWORD '123';
GRANT SELECT, INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA logistics
TO logistics_role;
GRANT SELECT, INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA inventory
TO logistics_role;
GRANT USAGE ON SCHEMA logistics TO logistics_role;
GRANT USAGE ON SCHEMA inventory TO logistics_role;

```

Приложение В

Код для заполнения таблицы

```

-- Fill tables with example data

INSERT INTO pdb.vendor (id, name, office, rating) VALUES (1, 'Hatber',
'Россия, г. Москва, ул. X1, дом Y1', 8);
INSERT INTO pdb.vendor (id, name, office, rating) VALUES (2, 'Самсон',
'Россия, г. Воронеж, ул. X2, дом Y2', 7);
INSERT INTO pdb.vendor (id, name, office, rating) VALUES (3, 'Апплика',
'Россия, г. Москва, ул. X3, дом Y3', 6);

INSERT INTO pdb.category (id, name) VALUES (1, 'Ручки');
INSERT INTO pdb.category (id, name) VALUES (2, 'Тетради');
INSERT INTO pdb.category (id, name) VALUES (3, 'Карандаши');

INSERT INTO pdb.product_type (code, vendor, category, raise) VALUES
(1001, 1, 1, 15);
INSERT INTO pdb.product_type (code, vendor, category, raise) VALUES
(1002, 2, 1, 10);

```

```
INSERT INTO pdb.product_type (code, vendor, category, raise) VALUES
(1003, 1, 2, 20);
```

```
INSERT INTO pdb.country (id, name) VALUES (1, 'Германия');
```

```
INSERT INTO pdb.country (id, name) VALUES (2, 'Россия');
```

```
INSERT INTO pdb.country (id, name) VALUES (3, 'Китай');
```

```
INSERT INTO staff.job (id, name, salary, days_per_week) VALUES (1,
'Менеджер', 5000, 4);
```

```
INSERT INTO staff.job (id, name, salary, days_per_week) VALUES (2,
'Кассир', 2500, 5);
```

```
INSERT INTO staff.job (id, name, salary, days_per_week) VALUES (3,
'Уборщик', 500, 7);
```

```
INSERT INTO staff.employee (id, job, name, active, exp_date) VALUES (1,
1, 'Джон Стоун', true, '2023-12-31');
```

```
INSERT INTO staff.employee (id, job, name, active, exp_date) VALUES (2,
2, 'Джек Воробей', true, '2024-06-30');
```

```
INSERT INTO staff.employee (id, job, name, active, exp_date) VALUES (3,
3, 'Партос', false, '2023-09-30');
```

```
INSERT INTO sales.customer (id, phone, name, reg_date) VALUES (1, 111,
'Вася Пупкин', '2022-01-15');
```

```
INSERT INTO sales.customer (id, phone, name, reg_date) VALUES (2, 121,
'Джек Ричер', '2022-02-28');
```

```
INSERT INTO sales.customer (id, phone, name, reg_date) VALUES (3, 131,
'Том Сойер', '2022-06-01');
```

```
INSERT INTO sales.sale (id, date, employee, customer, bonus, bonus_used)
VALUES (1, '2023-06-20 10:30:00', 1, 1, 50, 0);
```

```
INSERT INTO sales.sale (id, date, employee, customer, bonus, bonus_used)
VALUES (2, '2023-06-22 14:15:00', 2, 2, 25, 0);
```

```
INSERT INTO sales.sale (id, date, employee, customer, bonus, bonus_used)
VALUES (3, '2023-06-23 11:00:00', 3, 3, 75, 75);
```

```
INSERT INTO logistics.supply (id, date, done, employee, price) VALUES
(1, '2023-06-01', true, 1, 850);
```

```
INSERT INTO logistics.supply (id, date, done, employee, price) VALUES  
(2, '2023-06-15', true, 2, 750);  
INSERT INTO logistics.supply (id, date, done, employee, price) VALUES  
(3, '2023-06-30', false, 3, 1400);
```

```
INSERT INTO inventory.product (id, type, country, price, supply, sold,  
sale) VALUES (1, 1001, 1, 999, 1, false, null);  
INSERT INTO inventory.product (id, type, country, price, supply, sold,  
sale) VALUES (2, 1002, 2, 899, 2, true, 1);  
INSERT INTO inventory.product (id, type, country, price, supply, sold,  
sale) VALUES (3, 1003, 3, 1499, 3, false, null);  
INSERT INTO inventory.product (id, type, country, price, supply, sold,  
sale) VALUES (4, 1002, 2, 1299, 2, true, 2);  
INSERT INTO inventory.product (id, type, country, price, supply, sold,  
sale) VALUES (5, 1001, 3, 129, 3, true, 3);
```