НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування та спеціалізованих комп'ютерних систем

Лабораторна робота №3

з дисципліни **Бази даних і засоби управління**

на тему: "Засоби оптимізації роботи СУБД PostgreSQL"

Виконала: студентка 3 курсу

групи КВ-93

Баранова Є.В.

Перевірив:

Павловський В.І.

Київ – 2021

Постановка задачі

Метою роботи є здобуття практичних навичок використання засобів оптимізації СУБД PostgreSQL.

Загальне завдання роботи полягає у наступному:

1. Перетворити модуль “Модель” з шаблону MVC лабораторної роботи №2 у вигляд об’єктно-реляційної проекції (ORM).
2. Створити та проаналізувати різні типи індексів у PostgreSQL.
3. Розробити тригер бази даних PostgreSQL.
4. Навести приклади та проаналізувати рівні ізоляції транзакцій у PostgreSQL.

№ варіанта	*Види індексів*	*Умови для тригера*
3	*GIN, Hash*	*before delete, update*

Інформація про програму

Посилання на репозиторій у GitHub з вихідним кодом програми та звітом: https://github.com/BaranovaEugenia/DB_Lab3

Завдання №1

Модель «сутність-зв’язок» галузі продажу напоїв в кав’ярні.

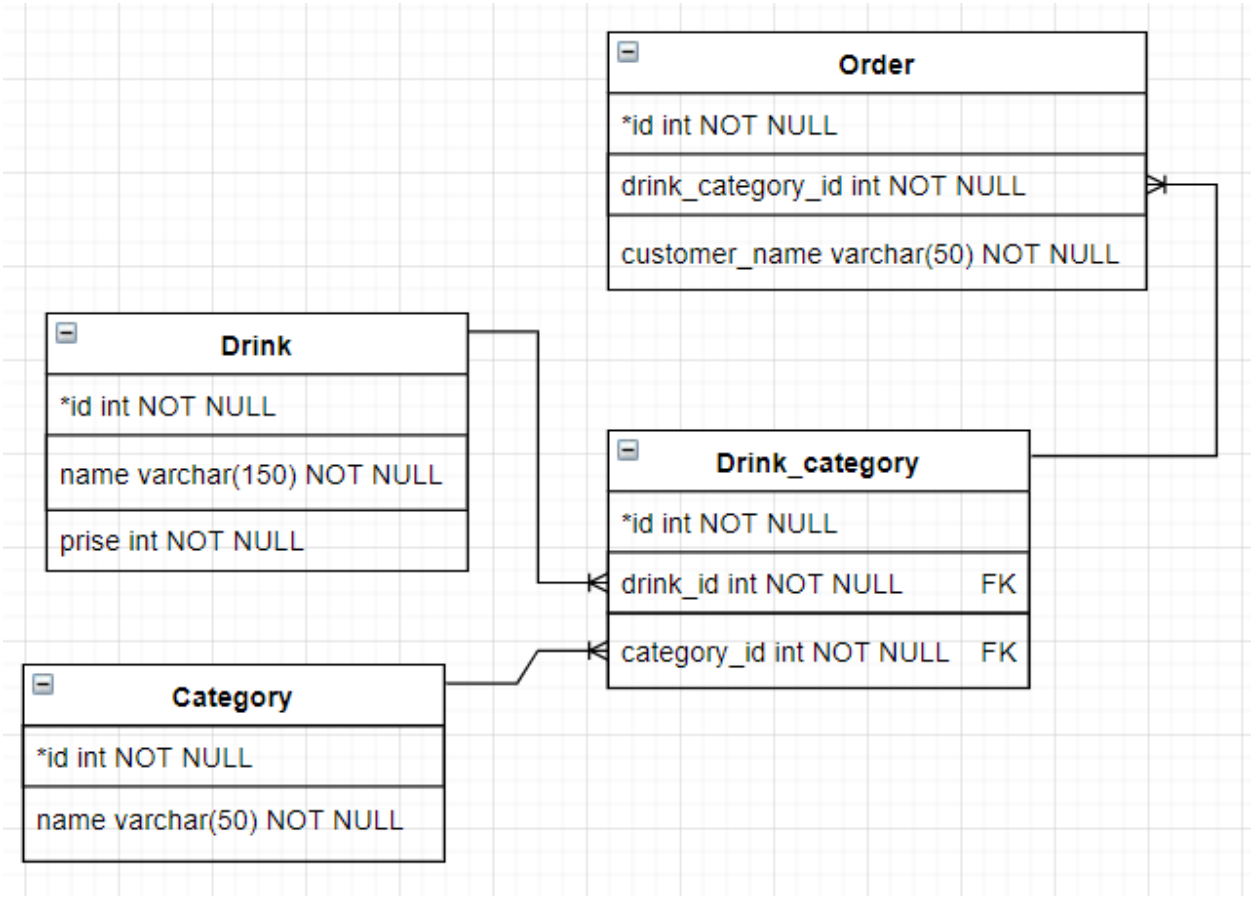

Рисунок 1 – Логічна модель

Таблиці бази даних у середовищі PgAdmin4

```
-- Table: public.Category

-- DROP TABLE IF EXISTS public."Category";

CREATE TABLE IF NOT EXISTS public."Category"
(
    id integer NOT NULL,
    name character varying(50) COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT category_pkey PRIMARY KEY (id)
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public."Category"
    OWNER to postgres;
```

```
-- Table: public.Drink
```

```
-- DROP TABLE IF EXISTS public."Drink";

CREATE TABLE IF NOT EXISTS public."Drink"
(
    id integer NOT NULL,
    name character varying(150) COLLATE pg_catalog."default" NOT NULL,
    "price" integer NOT NULL,
    CONSTRAINT drink_pkey PRIMARY KEY (id)
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public."Drink"
    OWNER to postgres;
```

```
-- Table: public.Drink_category

-- DROP TABLE IF EXISTS public."Drink_category";

CREATE TABLE IF NOT EXISTS public."Drink_category"
(
    id integer NOT NULL,
    drink_id integer NOT NULL,
    category_id integer NOT NULL,
    CONSTRAINT "customer _pkey" PRIMARY KEY (id),
    CONSTRAINT fk_category FOREIGN KEY (category_id)
        REFERENCES public."Category" (id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    CONSTRAINT fk_drink FOREIGN KEY (drink_id)
        REFERENCES public."Drink" (id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public."Drink_category"
    OWNER to postgres;
```

```
-- Table: public.Order

-- DROP TABLE IF EXISTS public."Order";

CREATE TABLE IF NOT EXISTS public."Order"
(
```

```
    id integer NOT NULL,
    "drink_category_id " integer NOT NULL,
    customer_name character varying(50) COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT order_pkey PRIMARY KEY (id),
    CONSTRAINT fk_drink_order FOREIGN KEY ("drink_category_id ")
        REFERENCES public."Drink_category" (id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public."Order"
    OWNER to postgres;
```

Класи ORM у реалізованому модулі Model

```
class Category(Orders):
    __tablename__ = 'Category'
    id = Column(Integer, primary_key=True)
    name = Column(String)

    def __init__(self, key, name):
        self.id = key
        self.name = name

    def __repr__(self):
        return "{:>10}{:>15}" \
            .format(self.id, self.name)
```

```
class Drink(Orders):
    __tablename__ = 'Drink'
    id = Column(Integer, primary_key=True)
    name = Column(String)
    price = Column(Integer)

    def __init__(self, key, name, price):
        self.id = key
        self.name = name
        self.price = price

    def __repr__(self):
        return "{:>10}{:>15}{:>10}" \
            .format(self.id, self.name, self.price)
```

```
class Drink_category(Orders):
    __tablename__ = 'Drink_category'
    id = Column(Integer, primary_key=True)
    category_id = Column(Integer, ForeignKey('Category.id'))
    drink_id = Column(Integer, ForeignKey('Drink.id'))
    categories = relationship("Category")
    drinks = relationship("Drink")

    def __init__(self, key, category_id, drink_id):
        self.id = key
        self.category_id = category_id
        self.drink_id = drink_id

    def __repr__(self):
        return "{:>10}{:>10}{:>10}" \
            .format(self.id, self.category_id, self.drink_id)
```

```
class Order(Orders):
    __tablename__ = 'Order'
    id = Column(Integer, primary_key=True)
    drink_category_id = Column(Integer, ForeignKey('Drink_category.id'))
    customer_name = Column(String)
    drink_categories = relationship("Drink_category")

    def __init__(self, key, drink_category_id, customer_name):
        self.id = key
        self.drink_category_id = drink_category_id
        self.customer_name = customer_name

    def __repr__(self):
        return "{:>10}{:>10}{:>15}" \
            .format(self.id, self.drink_category_id, self.customer_name)
```

Запити у вигляді ORM

Продемонструємо вставку, вилучення, редагування даних на прикладі таблиці

Drink

Початковий стан:

```
PS C:\Users\anna\PycharmProjects\pythonProject3> python main.py print_table Drink
Drink table:
        id           name     price
         1      Americano        30
         2       Espresso        30
         3  Hot chocolate        25
         4      Green_tea        15
         5          Cacao        25
         6          Latte        20
         7          gpjdb     41637
         8      lwctluvms     63794
```

Видалення запису:

```
PS C:\Users\anna\PycharmProjects\pythonProject3> python main.py delete_record Drink 7
PS C:\Users\anna\PycharmProjects\pythonProject3> python main.py print_table Drink
Drink table:
        id           name     price
         1      Americano        30
         2       Espresso        30
         3  Hot chocolate        25
         4      Green_tea        15
         5          Cacao        25
         6          Latte        20
         8      lwctluvms     63794
```

Вставка запису:

```
PS C:\Users\anna\PycharmProjects\pythonProject3> python main.py insert_record Drink 7 Lemonade 20
PS C:\Users\anna\PycharmProjects\pythonProject3> python main.py print_table Drink
Drink table:
        id           name     price
         1      Americano        30
         2       Espresso        30
         3  Hot chocolate        25
         4      Green_tea        15
         5          Cacao        25
         6          Latte        20
         7       Lemonade        20
         8      lwctluvms     63794
```

Редагування запису:

```
PS C:\Users\anna\PycharmProjects\pythonProject3> python main.py update_record Drink 8 Juice 25
PS C:\Users\anna\PycharmProjects\pythonProject3> python main.py print_table Drink
Drink table:
        id           name     price
         1      Americano        30
         2       Espresso        30
         3  Hot chocolate        25
         4      Green_tea        15
         5          Cacao        25
         6          Latte        20
         7       Lemonade        20
         8          Juice        25
```

Вставка 3-х випадково згенерованих записів:

```
PS C:\Users\anna\PycharmProjects\pythonProject3> python main.py generate_randomly Drink 3
PS C:\Users\anna\PycharmProjects\pythonProject3> python main.py print_table Drink
Drink table:
        id           name     price
         1      Americano        30
         2       Espresso        30
         3  Hot chocolate        25
         4      Green_tea        15
         5          Cacao        25
         6          Latte        20
         7       Lemonade        20
         8          Juice        25
         9        bhrylxj     36781
        10      glpfwokny     51439
        11        pupqdfx     52654
```

Усі дані для пошуку передвизначено, тепер вони не вводяться з клавіатури.

Пошук за трьома атрибутами у двох таблицях, за трьома атрибутами у трьох таблицях, за чотирма атрибутами у чотирьох таблицях (виводяться відповідні записи з таблиці Drink):

```
PS C:\Users\anna\PycharmProjects\pythonProject3> python main.py search_records
specify the number of tables you`d like to search in: 2
search result:
         4      Green_tea        15
         6          Latte        20
         7       Lemonade        20
```

```
PS C:\Users\anna\PycharmProjects\pythonProject3> python main.py search_records
specify the number of tables you`d like to search in: 3
search result:
        3  Hot chocolate        25
        5          Cacao        25
        4      Green_tea        15
        6          Latte        20
        7       Lemonade        20
        8          Juice        25
```

```
PS C:\Users\anna\PycharmProjects\pythonProject3> python main.py search_records
specify the number of tables you`d like to search in: 4
search result:
        2       Espresso        30
        1      Americano        30
```

Завдання №2

Для тестування індексів було створено окремі таблиці у базі даних з 1000000 записів.

Gin

Основна сфера застосування методу GIN - прискорення повнотекстового пошуку, тому логічно розглядати цей індекс більш докладно саме на цьому прикладі

Стверення таблиці БД:

```
DROP TABLE IF EXISTS "lab3_gin";
CREATE TABLE "lab3_gin"("id" serial PRIMARY KEY, "doc" text, "doc_tsv"
tsvector);
INSERT INTO "lab3_gin"("doc") SELECT chr((65 + round(random() * 25)) :: int) ||
chr((65 + round(random() * 25)) :: int)||chr((65 + round(random() * 25)) ::
int) FROM generate_series(1, 1000000);
UPDATE "lab3_gin" set "doc_tsv" = to_tsvector("doc");
```

Створення індексу:

```
CREATE INDEX "gin_i" ON "lab3_gin" USING gin("doc_tsv");
```

Видалення індексу:

```
DROP INDEX IF EXISTS "gin_i";
```

Запити SELECT

```
SELECT * FROM lab3_gin WHERE doc_tsv @@ to_tsquery('abc');
```

Data Output | Explain | Messages

	id [PK] integer	doc text	doc_tsv tsvector
1	31324	ABC	'abc':1
2	117510	ABC	'abc':1
3	236464	ABC	'abc':1
4	181499	ABC	'abc':1
5	120183	ABC	'abc':1
6	174842	ABC	'abc':1
7	259788	ABC	'abc':1
8	240328	ABC	'abc':1
9	298790	ABC	'abc':1
10	399851	ABC	'abc':1
11	334560	ABC	'abc':1
12	340617	ABC	'abc':1
13	359825	ABC	'abc':1

Data Output | Explain | Messages

	id [PK] integer	doc text	doc_tsv tsvector
14	427063	ABC	'abc':1
15	426641	ABC	'abc':1
16	569077	ABC	'abc':1
17	471661	ABC	'abc':1
18	511302	ABC	'abc':1
19	542378	ABC	'abc':1
20	488382	ABC	'abc':1
21	539318	ABC	'abc':1
22	551592	ABC	'abc':1
23	767510	ABC	'abc':1
24	619294	ABC	'abc':1
25	672298	ABC	'abc':1
26	760284	ABC	'abc':1

Data Output | Explain | Messages

	id [PK] integer	doc text	doc_tsv tsvector
22	551592	ABC	'abc':1
23	767510	ABC	'abc':1
24	619294	ABC	'abc':1
25	672298	ABC	'abc':1
26	760284	ABC	'abc':1
27	847085	ABC	'abc':1
28	823875	ABC	'abc':1
29	879331	ABC	'abc':1
30	854411	ABC	'abc':1
31	929802	ABC	'abc':1
32	938194	ABC	'abc':1
33	900282	ABC	'abc':1
34	977636	ABC	'abc':1

Час виконання запиту без використання GIN індексу

Data Output | Explain | Messages | Notifications

```
Successfully run. Total query runtime: 645 msec.
34 rows affected.
```

Час виконання запиту з використанням GIN індексу

Data Output | Explain | Messages | Notifications

```
Successfully run. Total query runtime: 70 msec.
34 rows affected.
```

```
SELECT COUNT(*) FROM lab3_gin WHERE doc_tsv @@ to_tsquery('xyz');
```

Data Output

	count bigint
1	35

Час виконання запиту без використання GIN індексу

Data Output Explain Messages Notifications

```
Successfully run. Total query runtime: 638 msec.
1 rows affected.
```

Час виконання запиту з використанням GIN індексу

Data Output Explain Messages Notifications

```
Successfully run. Total query runtime: 64 msec.
1 rows affected.
```

```
SELECT COUNT(*) FROM lab3_gin WHERE doc_tsv @@ to_tsquery('qwe') GROUP BY "id"
% 2;
```

Data Output

	count bigint
1	37
2	34

Час виконання запиту без використання GIN індексу

Data Output Explain Messages Notifications

```
Successfully run. Total query runtime: 500 msec.
2 rows affected.
```

Час виконання запиту з використанням GIN індексу

Data Output Explain Messages Notifications

```
Successfully run. Total query runtime: 75 msec.
2 rows affected.
```

```
SELECT * FROM lab3_gin WHERE doc_tsv @@ to_tsquery('qwe') ORDER BY RANDOM();
```

Data Output | Explain | Messages

	id [PK] integer	doc text	doc_tsv tsvector
1	769621	QWE	'qwe':1
2	447312	QWE	'qwe':1
3	425856	QWE	'qwe':1
4	196421	QWE	'qwe':1
5	878602	QWE	'qwe':1
6	63089	QWE	'qwe':1
7	983375	QWE	'qwe':1
8	657020	QWE	'qwe':1
9	662585	QWE	'qwe':1
10	705771	QWE	'qwe':1
11	802209	QWE	'qwe':1
12	632386	QWE	'qwe':1
13	369075	QWE	'qwe':1
14	583868	QWE	'qwe':1

Data Output | Explain | Messages

	id [PK] integer	doc text	doc_tsv tsvector
15	261942	QWE	'qwe':1
16	293714	QWE	'qwe':1
17	236996	QWE	'qwe':1
18	586935	QWE	'qwe':1
19	316693	QWE	'qwe':1
20	844066	QWE	'qwe':1
21	373586	QWE	'qwe':1
22	628276	QWE	'qwe':1
23	305676	QWE	'qwe':1
24	321527	QWE	'qwe':1
25	48028	QWE	'qwe':1
26	474177	QWE	'qwe':1
27	721802	QWE	'qwe':1
28	221884	QWE	'qwe':1

Час виконання запиту без використання GIN індексу

Data Output | Explain | Messages | Notifications

```
Successfully run. Total query runtime: 615 msec.
71 rows affected.
```

Час виконання запиту з використанням GIN індексу

Data Output | Explain | Messages | Notifications

```
Successfully run. Total query runtime: 78 msec.
71 rows affected.
```

З результатів видно, що використання GIN індексу, прискорює виконання запиту в 10 разів, це пояснюється тим, що до кожного елемента прив'язаний упорядкований набір посилань на рядки таблиці, що містять значення цього елемента.

Hash

Основна сфера застосування методу Hash - прискорення пошуку серед чисел , тому логічно розглядати цей індекс більш докладно саме на цьому прикладі.

Стверення таблиці БД:

```
DROP TABLE IF EXISTS "lab3_hash";
```

```
CREATE TABLE "lab3_hash"("id" serial PRIMARY KEY, "num" integer);
INSERT INTO "lab3_hash"("num") SELECT ((random() * 25) :: int) FROM
generate_series(1, 1000000);
```

Створення індексу:

```
CREATE INDEX "hash_i" ON "lab3_hash" USING hash("id");
```

Видалення індексу:

```
DROP INDEX IF EXISTS "hash_i";
```

Запити SELECT

```
SELECT * FROM lab3_hash WHERE num=25;
```

Data Output Explain Messages

	id [PK] integer	num integer
1	215	25
2	382	25
3	401	25
4	478	25
5	674	25
6	743	25
7	766	25
8	776	25
9	784	25
10	838	25
11	908	25
12	913	25
13	928	25

Час виконання запиту без використання Hash індексу

Data Output Explain Messages Notifications

```
Successfully run. Total query runtime: 218 msec.
20044 rows affected.
```

Час виконання запиту з використанням Hash індексу

Data Output Explain Messages Notifications

```
Successfully run. Total query runtime: 193 msec.
20044 rows affected.
```

```
SELECT COUNT(*) FROM lab3_hash WHERE num<10;
```

	count bigint
1	379987

Час виконання запиту без використання Hash індексу

Data Output Explain Messages Notifications

```
Successfully run. Total query runtime: 157 msec.
1 rows affected.
```

Час виконання запиту з використанням Hash індексу

Data Output Explain Messages Notifications

```
Successfully run. Total query runtime: 137 msec.
1 rows affected.
```

```
SELECT COUNT(*) FROM lab3_hash WHERE num=10 GROUP BY "id" % 2;
```

Data Output

	count bigint
1	20049
2	20297

Час виконання запиту без використання Hash індексу

Data Output Explain Messages Notifications

```
Successfully run. Total query runtime: 260 msec.
2 rows affected.
```

Час виконання запиту з використанням Hash індексу

Data Output Explain Messages Notifications

```
Successfully run. Total query runtime: 167 msec.
2 rows affected.
```

```
SELECT * FROM lab3_hash WHERE num=15 ORDER BY RANDOM();
```

Data Output | Explain | Messages

	id [PK] integer	num integer
1	773376	15
2	627834	15
3	519852	15
4	889342	15
5	185332	15
6	565909	15
7	205243	15
8	973259	15
9	923335	15
10	706461	15

Час виконання запиту без використання Hash індексу

Data Output | Explain | Messages | Notifications

```
Successfully run. Total query runtime: 177 msec.
40119 rows affected.
```

Час виконання запиту з використанням Hash індексу

Data Output | Explain | Messages | Notifications

```
Successfully run. Total query runtime: 165 msec.
40119 rows affected.
```

З отриманих результатів бачимо, що використання Hash індексу майже не покращує час виконання запиту, а якщо покращує то незначно. Ідея хешування полягає у тому, щоб значенню зіставити деяке невелике число (від 0 до N−1, всього N значень). Таке зіставлення називають хеш-функцією. Отримане число можна використовувати як індекс звичайного масиву, куди і складати посилання рядки таблиці.

Завдання №3

Суть роботи тригера полягає у тому, що у разі видалення чи оновлення рядків дорогих товарів в таблиці товарів, значення передаються в нову таблицю перевірки для того щоб працівник міг перевірити внесені зміни над дорогими товарами. При цьому якщо видаляти чи вносити зміни в дешеві товари, нічого не відбувається.

Створення тригерної функції:

```
CREATE OR REPLACE FUNCTION trigger_func() RETURNS TRIGGER as $trigger$
BEGIN
	IF old."price" >= 500 THEN
		INSERT INTO "lab3_trigger_for_check" VALUES (old.id, old.name,
		old.price, old.rest);
	END IF;
	IF (TG_OP = 'DELETE') THEN
		RETURN OLD;
	ELSIF (TG_OP = 'UPDATE') THEN
		RETURN NEW;
	END IF;
END;
$trigger$ LANGUAGE plpgsql;
```

Створення тригера:

```
CREATE TRIGGER "trigger1"
BEFORE DELETE OR UPDATE ON "lab3_trigger"
FOR EACH ROW
EXECUTE procedure trigger_func();
```

Початковий стан таблиці товарів та таблиці на перевірку

Query Editor | Query History

```
1 SELECT * FROM public.lab3_trigger
2 ORDER BY id ASC
```

Data Output | Explain | Messages | Notifications

	id [PK] integer	name character varying	price integer	rest integer
1	1	apple	5	100
2	2	gold	1000	3
3	3	fish	400	45
4	4	caviar	900	250
5	5	platinum	1100	5
6	6	water	3	300

Query Editor | Query History

```
1 SELECT * FROM public.lab3_trigger_for_check
2 ORDER BY id ASC
```

Data Output | Explain | Messages | Notifications

	id [PK] integer	name character varying	price integer	rest integer

Видалення дорогого товару

```
DELETE FROM "lab3_trigger" WHERE name='gold';
```

Data Output | Explain | Messages | Notifications

	id [PK] integer	name character varying	price integer	rest integer
1	1	apple	5	100
2	3	fish	400	45
3	4	caviar	900	250
4	5	platinum	1100	5
5	6	water	3	300

Data Output | Explain | Messages | Notifications

	id [PK] integer	name character varying	price integer	rest integer
1	2	gold	1000	3

Видалення дешевого товару

```
DELETE FROM "lab3_trigger" WHERE name='water';
```

Data Output Explain Messages Notifications

	id [PK] integer	name character varying	price integer	rest integer
1	1	apple	5	100
2	3	fish	400	45
3	4	caviar	900	250
4	5	platinum	1100	5

Data Output Explain Messages Notifications

	id [PK] integer	name character varying	price integer	rest integer
1	2	gold	1000	3

Оновлення дорогого товару

```
UPDATE "lab3_trigger" SET name='caviar', price=950, rest=249 WHERE id=4;
```

Data Output Explain Messages Notifications

	id [PK] integer	name character varying	price integer	rest integer
1	1	apple	5	100
2	3	fish	400	45
3	4	caviar	950	249
4	5	platinum	1100	5

Data Output Explain Messages Notifications

	id [PK] integer	name character varying	price integer	rest integer
1	2	gold	1000	3
2	4	caviar	950	249

Оновлення дешевого товару

```
UPDATE "lab3_trigger" SET name='caviar', price=950, rest=249 WHERE id=4;
```

Data Output Explain Messages Notifications

	id [PK] integer	name character varying	price integer	rest integer
1	1	apple	4	200
2	3	fish	400	45
3	4	caviar	950	249
4	5	platinum	1100	5

Data Output Explain Messages Notifications

	id [PK] integer	name character varying	price integer	rest integer
1	2	gold	1000	3
2	4	caviar	900	250