

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”*

Виконав:

студент ІII курсу

групи КВ-01

Черичка М. А.

Перевірив: Павловський В. І.

Київ – 2022

**Варіант 21**

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.

## Логічна модель бази даних

Нижче (Рисунок 1) наведено логічну модель бази даних:

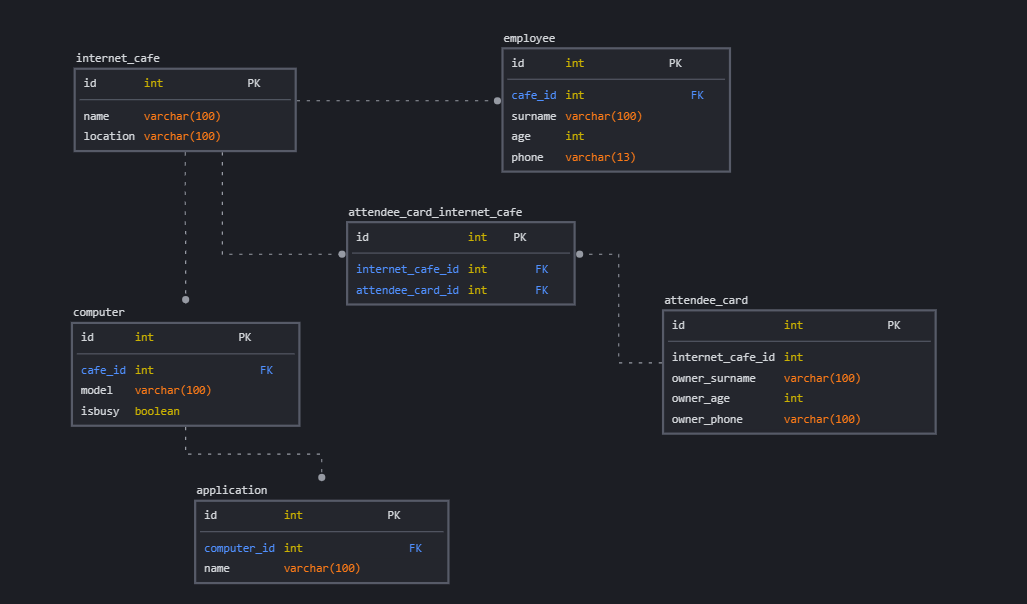


Рисунок 1 – логічна модель бази даних

Зміни у порівнянні з першою лабораторною роботою відсутні.

## Класи ORM

class internet\_cafe(Base):  
 \_\_tablename\_\_ = 'internet\_cafe'  
 id = Column(Integer, primary\_key=True,)  
 name = Column(String)  
 location = Column(String)  
  
 def \_\_init\_\_(self, name, location):  
 self.name = name  
 self.location = location  
 super(internet\_cafe, self).\_\_init\_\_()  
  
class employee(Base):  
 \_\_tablename\_\_ = 'employee'  
 id = Column(Integer, primary\_key=True)  
 cafe\_id = Column(Integer, ForeignKey('internet\_cafe.id'))  
 surname = Column(String)  
 age = Column(Integer)  
 phone = Column(String)  
  
 def \_\_init\_\_(self, cafe\_id, surname, age, phone):  
 self.cafe\_id = cafe\_id  
 self.surname = surname  
 self.age = age  
 self.phone = phone  
 super(employee, self).\_\_init\_\_()  
  
class computer(Base):  
 \_\_tablename\_\_ = 'computer'  
 id = Column(Integer, primary\_key=True)  
 cafe\_id = Column(Integer, ForeignKey('internet\_cafe.id'))  
 model = Column(String)  
 isbusy = Column(Boolean)  
  
 def \_\_init\_\_(self, cafe\_id, model, isbusy):  
 self.cafe\_id = cafe\_id  
 self.model = model  
 self.isbusy = isbusy  
 super(computer, self).\_\_init\_\_()  
  
class application(Base):  
 \_\_tablename\_\_ = 'application'  
 id = Column(Integer, primary\_key=True)  
 computer\_id = Column(Integer, ForeignKey('computer.id'))  
 name = Column(String)  
  
 def \_\_init\_\_(self, computer\_id, name):  
 self.computer\_id = computer\_id  
 self.name = name  
 super(application, self).\_\_init\_\_()  
  
class attendee\_card\_internet\_cafe(Base):  
 \_\_tablename\_\_ = 'attendee\_card\_internet\_cafe'  
 id = Column(Integer, primary\_key=True)  
 internet\_cafe\_id = Column(Integer, ForeignKey('internet\_cafe.id'))  
 attendee\_card\_id = Column(Integer, ForeignKey('attendee\_card.id'))  
  
 def \_\_init\_\_(self, internet\_cafe\_id, attendee\_card\_id):  
 self.internet\_cafe\_id = internet\_cafe\_id  
 self.attendee\_card\_id = attendee\_card\_id  
 super(attendee\_card\_internet\_cafe, self).\_\_init\_\_()

class attendee\_card(Base):  
 \_\_tablename\_\_ = 'attendee\_card'  
 id = Column(Integer, primary\_key=True)  
 internet\_cafe\_id = Column(Integer, ForeignKey('internet\_cafe.id'))  
 owner\_surname = Column(String)  
 owner\_age = Column(Integer)  
 owner\_phone = Column(String)  
  
 def \_\_init\_\_(self, internet\_cafe\_id, owner\_surname, owner\_age, owner\_phone):  
 self.internet\_cafe\_id = internet\_cafe\_id  
 self.owner\_surname = owner\_surname  
 self.owner\_age = owner\_age  
 self.owner\_phone = owner\_phone  
 super(attendee\_card, self).\_\_init\_\_()

**Тригер бази даних PostgreSQL. Умова для тригера – before delete, update**

**Таблиці:**

CREATE TABLE employee (

id integer NOT NULL GENERATED ALWAYS AS IDENTITY ( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),

username varchar NOT NULL,

salary integer NOT NULL,

last\_update timestamp

);

INSERT INTO employee (username, salary, last\_update)

VALUES ('Billy', 1000, current\_timestamp);

CREATE TABLE employee\_audit (

operation varchar(10) NOT NULL,

stamp timestamp NOT NULL,

employee\_id integer NOT NULL,

username varchar NOT NULL,

salary integer NOT NULL

);

**Функція:**

CREATE OR REPLACE FUNCTION employee\_stamp() RETURNS trigger AS $employee\_stamp$

BEGIN

IF (TG\_OP = 'DELETE') THEN

INSERT INTO employee\_audit SELECT 'DELETE', now(), OLD.id, OLD.username, OLD.salary;

ELSIF (TG\_OP = 'UPDATE') THEN

IF NEW.salary < 0 THEN

RAISE EXCEPTION '% cannot have a negative salary', NEW.username;

END IF;

INSERT INTO employee\_audit SELECT 'UPDATE', now(), NEW.id, NEW.username, NEW.salary;

END IF;

NEW.last\_update := current\_timestamp;

RETURN NEW;

END;

$employee\_stamp$ LANGUAGE plpgsql;

CREATE TRIGGER employee\_stamp BEFORE DELETE OR UPDATE ON employee

FOR EACH ROW EXECUTE FUNCTION employee\_stamp();

## Принцип роботи тригеру

Тригер спрацьовує перед видаленням або редагуванням в таблиці «employee». При редагуванні або видаленні данних з таблиці «employee» у таблицю «employee\_audit» буде заноситись рядок про тип операції, часова мітка виконання та дані про робітника. Якщо обрана операція була UPDATE, тоді ще проводиться додаткова перевірка на ввід некоректних даних (зарплатня не може бути менше 0).

***Приклад***

**Таблиці до змін:**

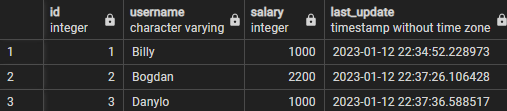


Рисунок 2 – вигляд таблиці “employee”.

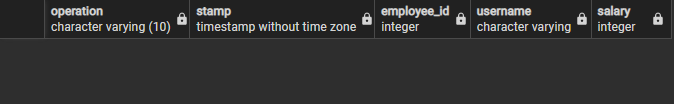


Рисунок 3 – вигляд таблиці “employee\_audit”.

**Виконаємо оновлення одного рядка:**

UPDATE employee

SET salary = 1500

WHERE id = 1;

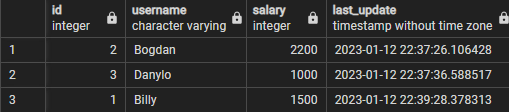


Рисунок 4 – вигляд таблиці “employee” після оновлення одного рядка.

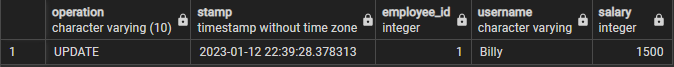


Рисунок 5 – вигляд таблиці “employee\_audit” після оновлення одного рядка в таблиці “employee”.

**Спробуємо задати некоректні дані:**

UPDATE employee

SET salary = -1500

WHERE id = 1;

Отримуємо помилку, а самі таблиці ніяк не змінились

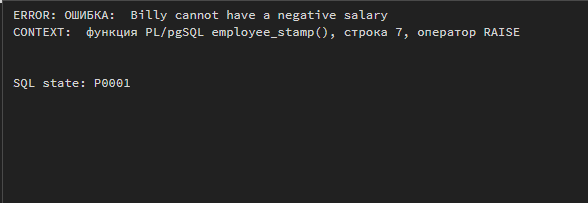


Рисунок 6 – вигляд виводу помилки після спроби задання некоректних даних.

**Виконаємо видалення рядка:**

DELETE FROM employee

WHERE id = 1;



Рисунок 7 – вигляд таблиці “employee” після видалення одного рядка.

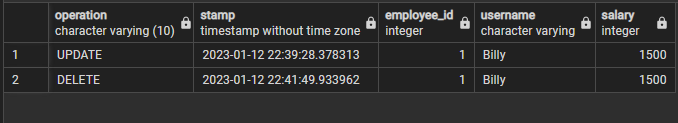


Рисунок 8 – вигляд таблиці “employee\_audit” після видалення одного рядка з таблиці “employee”.