BAZA DANYCH DLA BIBLIOTEKI - PIOTR LEŻAŃSKI

Wstępny opis projektu

Mój projekt polega na zaimplementowaniu tabel oraz umożliwienia wykonywania na nich podstawowych operacji, których działanie ma symulować operacje wykonywane w bibliotece książek. Program został napisany przy użyciu biblioteki Tkinter języka Python oraz języka SQL w standardzie PostgreSQL.

Mój projekt przyjmuje formę aplikacji okienkowej, w której użytkownik ma możliwość wykonania poniższych operacji:

1) Wyświetlenie wszystkich książek w bibliotece („View All”)
2) Wyszukanie książki za pomocą tytułu („Search Title”)
3) Otworzenie linku, który przekieruje użytkownika do przeglądarki z wpisanym hasłem w postaci tytułu zaznaczonej książki („Open Link”)
4) Zmiana tytułu zaznaczonej książki („Update Title”)
5) Usunięcie z bazy danych zaznaczonej książki („Delete Entry”)

Plik o nazwie frontend.py posiada kompletny kod potrzebny do wygenerowania frontendu (wyglądu) mojej aplikacji. Plik o nazwie backend.py posiada opis kwerend z języka SQL, czyli działań wykonywanych na danej bazie danych.

W pliku sql_queries.sql znajduje się kompletny spis wszystkich wymaganych kwerend. Ich opis znajduje się w dalszej części projektu.

Ograniczeniem programu jest fakt, że nie jest możliwe dodanie nowej książki/autora itp. Aplikacja napisana przeze mnie obsługuje jedynie zapytania dokonane przez użytkownika. Dla administratora (bibliotekarza/rki) konieczne by było stworzenie drugiej wersji programu.

OPIS TABEL

Authors

Tabela przechowuje autorów oraz informacje o nich.

Association

Przechowuje informacje o stowarzyszeniach, które zrzeszają autorów.

Authors_territories

Przechowuje id autora oraz id regionu, z którego pochodzi.

Territories

Przechowuje id oraz nazwę regionu.

Books

Tabela przechowuje książki oraz informacje o nich.

Types

Przechowuje nazwy typów książek.

Suppliers

Przechowuje informacje o nazwie firm wydających książki.

Links

Przechowuje linki do książek.

Orders

Przechowuje informacje o zamówieniach.

Customers

Przechowuje informacje o klientach.

SKRYPT TWORZĄCY BAZĘ DANYCH

```
CREATE TABLE Authors (
    author_id INT,
    name VARCHAR(20),
    last_name VARCHAR(25),
    address VARCHAR(60),
    association_id INT,
    territory_id INT,
    PRIMARY KEY(author_id),
    FOREIGN KEY(association_id) REFERENCES Association(association_id) ON DELETE CASCADE,
    FOREIGN KEY(territory_id) REFERENCES Authors_territories(territory_id)
);

CREATE TABLE Authors_territories (
    author_id INT UNIQUE,
    territory_id INT,
    PRIMARY KEY(author_id),
    FOREIGN KEY(territory_id) REFERENCES Territories(territory_id) ON DELETE CASCADE
);

CREATE TABLE Types (
    type_id INT,
    type_name VARCHAR(30),
    PRIMARY KEY(type_id)
);

CREATE TABLE Territories (
    territory_id INT,
    territory_name VARCHAR(30),
    PRIMARY KEY(territory_id)
);

CREATE TABLE Association (
    association_id INT,
    name VARCHAR(40),
    members INT,
    PRIMARY KEY(association_id)
);

CREATE TABLE Books (
    book_id INT,
    title VARCHAR(40),
    author_id INT,
    type_id INT,
    supplier_id INT,
    PRIMARY KEY(book_id),
    FOREIGN KEY(author_id) REFERENCES Authors(author_id) ON DELETE CASCADE,
    FOREIGN KEY(type_id) REFERENCES Types(type_id) ON DELETE CASCADE,
    FOREIGN KEY(supplier_id) REFERENCES Suppliers(supplier_id) ON DELETE CASCADE
);
```

SKRYPT TWORZĄCY BAZĘ DANYCH

```
CREATE TABLE Suppliers (
    supplier_id INT,
    company_name VARCHAR(30),
    region VARCHAR(20),
    phone VARCHAR(20),
    PRIMARY KEY(supplier_id)
);

CREATE TABLE Links (
    book_id INT,
    link VARCHAR(80),
    PRIMARY KEY(book_id)
);

CREATE TABLE Orders (
    order_id INT,
    customer_id INT,
    unit_price INT,
    quantity INT,
    PRIMARY KEY(order_id),
    FOREIGN KEY(customer_id) REFERENCES Customers(customer_id) ON DELETE CASCADE
);

CREATE TABLE Customers (
    customer_id INT,
    customer_name VARCHAR(30),
    PRIMARY KEY(customer_id)
);
```

```
INSERT INTO Authors VALUES
    (1, 'James', 'Patterson', '777 Brockton Avenue, Abington MA 2351', 4, 2),
    (2, 'Elly', 'Griffiths', '30 Memorial Drive, Avon MA 2322', 4, 1),
    (3, 'Lj', 'Ross', '250 Hartford Avenue, Bellingham MA 2019', 2, 3),
    (4, 'Lee', 'Child', '700 Oak Street, Brockton MA 2301', 1, 4),
    (5, 'Michael', 'Connelly', '66-4 Parkhurst Rd, Chelmsford MA 1824', 3, 2),
    (6, 'Ann', 'Cleves', '591 Memorial Dr, Chicopee MA 1020', 1, 1),
    (7, 'Peter', 'James', '55 Brooksby Village Way, Danvers MA 1923', 6, 5),
    (8, 'JD', 'Kirk', '137 Teaticket Hwy, East Falmouth MA 2536', 5, 5),
    (9, 'Bernard', 'Cornwell', '42 Fairhaven Commons Way, Fairhaven MA 2719', 6, 1),
    (10, 'Louise', 'Penny', '374 William S Canning Blvd, Fall River MA 2721', 4, 6),
    (11, 'Dilly', 'Count', '121 Worcester Rd, Framingham MA 1701', 5, 2),
    (12, 'Stuard', 'MacBridge', '677 Timpany Blvd, Gardner MA 1440', 7, 1),
    (13, 'Ian', 'Rankin', '337 Russell St, Hadley MA 1035', 3, 1),
    (14, 'Nora', 'Roberts', '295 Plymouth Street, Halifax MA 2338', 5, 8),
    (15, 'Clive', 'Cussler', '1775 Washington St, Hanover MA 2339', 1, 7),
    (16, 'Anna', 'Jacobs', '280 Washington Street, Hudson MA 1749', 6, 7),
    (17, 'D j', 'Sansom', '20 Soojian Dr, Leicester MA 1524', 3, 8),
    (18, 'Mick', 'Heron', '11 Jungle Road, Leominster MA 1453', 2, 3),
    (19, 'Lj', 'Ross', '301 Massachusetts Ave, Lunenburg MA 1462', 4, 2),
    (20, 'Lj', 'Ross', '780 Lynnway, Lynn MA 1905', 7, 9);

INSERT INTO Authors_territories VALUES
    (1, 5),
    (2, 9),
    (3, 4),
    (4, 1),
    (5, 2),
    (6, 6),
    (7, 2),
    (8, 4),
    (9, 8),
    (10, 7),
    (11, 9),
    (12, 2),
    (13, 1),
    (14, 1),
    (15, 5),
    (16, 3),
    (17, 2),
    (18, 4),
    (19, 7),
    (20, 6);
```

SKRYPT TWORZĄCY BAZĘ DANYCH

```
INSERT INTO Association VALUES
    (1, 'Academy of American Poets', 350),
    (2, 'Society of Indexers ', 245),
    (3, 'Authors Guild, The', 130),
    (4, 'Canadian Authors Association', 400),
    (5, 'Mystery Writers of America', 120),
    (6, 'National Writers Union', 35),
    (7, 'Novelists, Inc.', 260);

INSERT INTO Territories VALUES
    (1, 'Belgium'),
    (2, 'North America'),
    (3, 'Poland'),
    (4, 'Japan'),
    (5, 'Croatia'),
    (6, 'Netherlands'),
    (7, 'Scotland'),
    (8, 'Germany'),
    (9, 'Sweden');

INSERT INTO Types VALUES
    (1, 'Crime'),
    (2, 'Fiction'),
    (3, 'Thriller'),
    (4, 'Fantasy'),
    (5, 'Romance'),
    (6, 'Adventure'),
    (7, 'Childern''s Fiction'),
    (8, 'Scientific');

INSERT INTO Suppliers VALUES
    (1, 'Transworld', 'North America', '385910938'),
    (2, 'Bloomsbury', 'Sweden', '143643673'),
    (3, 'Random House', 'North America', '13573523'),
    (4, 'Brown Book', 'Germany', '63423575'),
    (5, 'Quercus', 'North America', '6247352'),
    (6, 'Macmillan', 'Mexico', '26752426'),
    (7, 'Penguin', 'North America', '24675235'),
    (8, 'Hodder', 'Poland', '24657323');
```

SKRYPT TWORZĄCY BAZĘ DANYCH

```sql
INSERT INTO Books VALUES
    (1, 'Da Vinci Code', 18, 1, 7),
    (2, 'Harry Potter and Deathly Hallows', 10, 4, 4),
    (3, 'Fifty Shades of Grey', 1, 6, 3),
    (4, 'Angles and Demons', 2, 8, 2),
    (5, 'Twilight', 2, 4, 2),
    (6, 'New Moon', 1, 2, 4),
    (7, 'The Art of War', 18, 3, 1),
    (8, 'Hostage at The Table', 14, 4, 4),
    (9, 'Why', 14, 1, 3),
    (10, 'Cryptography', 12, 2, 7),
    (11, 'Short History of Time', 11, 2, 7),
    (12, 'Tango', 13, 7, 6),
    (13, 'Rich Dad, Poor Dad', 3, 7, 8),
    (14, 'Eclipse', 18, 6, 3),
    (15, 'Lovely Bones', 9, 5, 4),
    (16, 'Fortress', 5, 5, 6),
    (17, 'Breaking Down', 18, 6, 3),
    (18, 'Gruffalo', 19, 8, 2),
    (19, 'One Day', 18, 3, 1),
    (20, 'Atonement', 3, 2, 6),
    (21, 'Sound of Laughter', 19, 1, 7),
    (22, 'Life of Pi', 17, 2, 8),
    (23, 'Birdsong', 20, 2, 2),
    (24, 'Labyrinth', 3, 8, 2),
    (25, 'Help', 1, 6, 4),
    (26, 'Memoirs', 8, 4, 5),
    (27, 'Island', 12, 4, 6),
    (28, 'Broker', 11, 5, 5);

INSERT INTO Links VALUES
    (1, 'https://www.google.com/search?q=da+vinci+code+book'),
    (2, 'https://www.google.com/search?q=harry+potter+and+deathly+hallows'),
    (3, 'https://www.google.com/search?q=fifty+shades+of+grey'),
    (4, 'https://www.google.com/search?q=angels+and+demons+book'),
    (5, 'https://www.google.com/search?q=twilight+book'),
    (6, 'https://www.google.com/search?q=new+moon'),
    (7, 'https://www.google.com/search?q=the+art+of+war'),
    (8, 'https://www.google.com/search?q=hostage+at+the+table'),
    (9, 'https://www.google.com/search?q=why+book'),
    (10, 'https://www.google.com/search?q=cryptography+book'),
    (11, 'https://www.google.com/search?q=short+history+of+time'),
    (12, 'https://www.google.com/search?q=tango+book'),
    (13, 'https://www.google.com/search?q=rich+dad+poor+dad'),
    (14, 'https://www.google.com/search?q=eclipse+book'),
    (15, 'https://www.google.com/search?q=lovely+bones+book'),
    (16, 'https://www.google.com/search?q=fortress+book'),
    (17, 'https://www.google.com/search?q=breaking+down+book'),
    (18, 'https://www.google.com/search?q=gruffalo+book'),
    (19, 'https://www.google.com/search?q=one+day+book'),
    (20, 'https://www.google.com/search?q=atonement+book'),
    (21, 'https://www.google.com/search?q=sound+of+laughter+book'),
    (22, 'https://www.google.com/search?q=life+of+pi+book'),
    (23, 'https://www.google.com/search?q=birdsong+book'),
    (24, 'https://www.google.com/search?q=labyrinth+book'),
    (25, 'https://www.google.com/search?q=help+book'),
    (26, 'https://www.google.com/search?q=memoirs+book'),
    (27, 'https://www.google.com/search?q=island+book'),
    (28, 'https://www.google.com/search?q=broker+book');
```

SKRYPT TWORZĄCY BAZĘ DANYCH

```sql
INSERT INTO Orders VALUES
    (1, 4, 54, 635),
    (2, 1, 34, 253),
    (3, 2, 2, 653),
    (4, 2, 4, 8653),
    (5, 5, 14, 153),
    (6, 3, 56, 9053),
    (7, 5, 75, 3335),
    (8, 8, 12, 1235),
    (9, 8, 24, 1653),
    (10, 3, 13, 4556),
    (11, 3, 64, 6213),
    (12, 2, 13, 22442),
    (13, 1, 73, 2964),
    (14, 2, 86, 313),
    (15, 5, 24, 133),
    (16, 7, 526, 251),
    (17, 8, 64, 9953),
    (18, 8, 264, 4525),
    (19, 2, 75, 6133),
    (20, 1, 34, 6515);

INSERT INTO Customers VALUES
    (1, 'Simon & Schuster'),
    (2, 'HarperCollins'),
    (3, 'Macmillan Publishers'),
    (4, 'Penguin Random House'),
    (5, 'Hachette Livre'),
    (6, 'Scholastic'),
    (7, 'Pearson Education'),
    (8, 'McGraw-Hill Education');
```

WIDOKI I FUNKCJE

Widok 1

Wypisuje klientów, którzy nie złożyli żadnego zamówienia

```
CREATE VIEW customers_no_order AS
    SELECT c.customer_id, c.customer_name FROM Customers c
    LEFT JOIN Orders o ON(o.customer_id = c.customer_id)
    WHERE order_id IS NULL
```

Widok 2

Wypisuje autorów z Niemiec

```
CREATE VIEW from_germany AS
    SELECT A.author_id, name FROM Authors A
    JOIN Authors_territories At ON(A.author_id = At.author_id)
    JOIN Territories T ON(At.territory_id = T.territory_id)
    WHERE territory_name = 'Germany';
```

Widok 3

Wypisuje książki przygodowe.

```
CREATE VIEW adventure_books AS
    SELECT book_id, title FROM Books B
    JOIN Types T ON(B.type_id = T.type_id)
    WHERE T.type_name = 'Adventure';
```

Widok 4

Wypisuje dostawcę z największą ilością wydanych książek.

```
CREATE VIEW max_supplier AS
    SELECT company_name FROM Books B
    JOIN Suppliers S ON(B.supplier_id = S.supplier_id)
    GROUP BY company_name
    ORDER BY count(*) DESC
    LIMIT 1;
```

Widok 5

Wypisuje stowarzyszenie z największą liczbą członków.

```
CREATE VIEW max_association AS
    SELECT Ass.name, count(*) AS num_of_members FROM Association Ass
    JOIN Authors A ON(Ass.association_id = A.association_id)
    GROUP BY Ass.name
    ORDER BY count(*) DESC
    LIMIT 1
```

Widok 6

Wypisuje autorów, którzy nie należą do żadnego stowarzyszenia.

```
CREATE VIEW no_association AS
    SELECT author_id FROM Authors A
    LEFT JOIN Association Ass
    ON(A.association_id = Ass.association_id)
    WHERE Ass.association_id IS NULL
```

Widok 7

Wypisuje autora oraz wydawce jego książki.

```
CREATE VIEW author_supplier AS
    SELECT A.name, A.last_name, S.company_name FROM Authors A
    JOIN Books B ON(A.author_id = B.author_id)
    JOIN Suppliers S ON(B.supplier_id = S.supplier_id)
```

Widok 8

Wypisuje książki napisane przez autorów ze Szkocji.

```
CREATE VIEW books_scotland AS
    SELECT B.title FROM Books B
    JOIN Authors A ON(B.author_id = A.author_id)
    JOIN Authors_territories At ON (A.author_id = At.author_id)
    JOIN Territories T ON(At.territory_id = T.territory_id)
    WHERE T.territory_name = 'Scotland';
```

Widok 9

Wypisuje zamówienia z największą kwotą.

```
CREATE VIEW max_order AS
    SELECT order_id, (unit_price * quantity) AS cost
    FROM Orders
    ORDER BY cost DESC
    LIMIT 1;
```

Funkcja

Wypisuje wypisuje wszystkie książki podanego typu. Funkcja zwraca tabelę, w której znajdować się będą wyniki. Jako argument przyjmuje stringa mówiącego o typie książek, jakie szukamy. Jeżeli podamy typ, który nie znajduje się w tabeli Types, wynikowa tabela będzie pusta.

```
CREATE OR REPLACE FUNCTION books_from_type(type_n VARCHAR)
    RETURNS TABLE (
        title VARCHAR
    )
AS $$
BEGIN
    RETURN QUERY
    SELECT B.title FROM Books B
    JOIN Types T ON(B.type_id = T.type_id)
    WHERE T.type_name = type_n;
END; $$ LANGUAGE 'plpgsql';
```

WYZWALACZE

```
CREATE TABLE Audit(
    command VARCHAR(40)
);
```

Wyzwalacz 1

Dodawanie informacji o nowo dodanej książce. Funkcja book_insert jest bezargumentowa. Zwraca wyzwalacz, który do tabeli Audit dodaje stosowny komunikat. Wyzwalacz zostaje wywołany przed operacją INSERT do tabeli Books.

```
CREATE FUNCTION book_insert()
    RETURNS TRIGGER
    LANGUAGE 'plpgsql'
AS $$
    BEGIN
        INSERT INTO Audit VALUES ('New book added');
        RETURN NEW;
END; $$

CREATE TRIGGER book_added
    BEFORE INSERT
    ON Books
        FOR EACH ROW
            EXECUTE PROCEDURE book_insert()
```

Wyzwalacz 2

Dodawanie id typu nowo dodanej książki. Funkcja book_type jest bezargumentowa. Zwraca wyzwalacz, który do tabeli Audit dodaje id. Wyzwalacz zostaje wywołany przed operacją INSERT do tabeli Books.

```
CREATE FUNCTION book_type()
    RETURNS TRIGGER
    LANGUAGE 'plpgsql'
AS $$
    BEGIN
        INSERT INTO Audit VALUES (NEW.type_id);
        RETURN NEW;
END; $$

CREATE TRIGGER book_added_type
    BEFORE INSERT
    ON Books
        FOR EACH ROW
            EXECUTE PROCEDURE book_type()
```

Wyzwalacz 3

Funkcja author_added jest bezargumentowa. Zwraca wyzwalacz, który do tabeli Audit dodaje imię i nazwisko nowo dodanego autora. Wyzwalacz zostaje wywołany przed operacją INSERT do tabeli Authors.

```
CREATE FUNCTION author_added()
    RETURNS TRIGGER
    LANGUAGE 'plpgsql'
AS $$
    BEGIN
        INSERT INTO Audit VALUES (NEW.name + NEW.last_name);
        RETURN NEW;
END; $$

CREATE TRIGGER author_added
    BEFORE INSERT
    ON Authors
        FOR EACH ROW
            EXECUTE PROCEDURE author_added()
```

Wyzwalacz 4

Funkcja name_updated jest bezargumentowa. Zwraca wyzwalacz, który do tabeli Audit dodaje starą nazwę firmy wraz z komunikatem, na jaką nazwę została ona zmieniona. Wyzwalacz zostaje wywołany przed operacją UPDATE do tabeli Suppliers.

```
CREATE FUNCTION name_updated()
    RETURNS TRIGGER
    LANGUAGE 'plpgsql'
AS $$
    BEGIN
        INSERT INTO Audit VALUES (OLD.company_name + ' changed to ' + NEW.company_name);
        RETURN NEW;
END; $$

CREATE TRIGGER name_changed
    BEFORE UPDATE
    ON Suppliers
        FOR EACH ROW
            EXECUTE PROCEDURE name_updated();
```

Wyzwalacz 5

Funkcja author_deleted jest bezargumentowa. Zwraca wyzwalacz, który do tabeli Audit dodaje imię usuniętego właśnie autora. Wyzwalacz zostaje wywołany przed operacją DELETE na tabeli Authors.

```
CREATE FUNCTION author_deleted()
    RETURNS TRIGGER
    LANGUAGE 'plpgsql'
AS $$
    BEGIN
        INSERT INTO Audit VALUES (OLD.name + ' deleted');
        RETURN NEW;
END; $$

CREATE TRIGGER author_delete
    BEFORE DELETE
    ON Authors
        FOR EACH ROW
            EXECUTE PROCEDURE author_deleted();
```

PROCEDURY

Procedura 1

Dodawanie nowego linku do książki. Procedura jako argumenty dostaje id książki oraz nowy link do zmiany. Ustawia ona nowy link w miejsce starego.

```
CREATE OR REPLACE PROCEDURE change_link(b_id INT, new_link VARCHAR)
LANGUAGE 'plpgsql'
AS $$
    BEGIN
        UPDATE Links
        SET link = new_link
        WHERE book_id = b_id;
        COMMIT;
END; $$
```

Procedura 2

Dodawanie nowego członka do stowarzyszenia. Procedura otrzymuje jako argument id stowarzyszenia oraz zwiększa ilość jego członków o jeden.

```
CREATE OR REPLACE PROCEDURE memeber_add(ass_id int)
LANGUAGE 'plpgsql'
AS $$
    BEGIN
        UPDATE Association
        SET memebers = members + 1
        WHERE association_id = ass_id;
        COMMIT;
END; $$
```

Procedura 3

Zmienianie wszystkich książek typu1 na typ2. Procedura otrzymuje jako argumenty typ książek, które należy zmienić by były typu drugiego.

```
CREATE OR REPLACE PROCEDURE change_link(type1 VARCHAR, type2 VARCHAR)
LANGUAGE 'plpgsql'
AS $$
    BEGIN
        UPDATE Books
        SET type_id = (SELECT type_id FROM Types
                       WHERE type_name = type2)
        WHERE type_id = (SELECT type_id FROM Types
                         WHERE type_name = type1);
        COMMIT;
END; $$
```

Procedura 4

Zmniejszanie ceny jednostkowej o wartość value_d, w rekordach, w których customer_name to cus_name.

```
CREATE OR REPLACE PROCEDURE change_link(value_d INT, cus_name VARCHAR)
LANGUAGE 'plpgsql'
AS $$
    BEGIN
        UPDATE Orders
        SET unit_price =
            CASE
                WHEN unit_price <= value_d THEN
                    unit_price = 0
                ELSE unit_price = unit_price - value_d
            END
        WHERE customer_id = (
            SELECT customer_id
            FROM Customers
            WHERE customer_name = cus_name
        );
        COMMIT;
END; $$
```

Procedura 5

Wszystkie książki o typie type1 od teraz są obsługiwane przez wydawce o nazwie sup_name.

```
CREATE OR REPLACE PROCEDURE change_supp(type1 VARCHAR, sup_name VARCHAR)
LANGUAGE 'plpgsql'
AS $$
    BEGIN
        UPDATE Books
        SET supplier_id = (SELECT supplier_id
                           FROM Supplier
                           WHERE supplier_name = sup_name)
        WHERE type_id = (SELECT type_id
                         FROM Types
                         WHERE type_name = type1);
        COMMIT;
END; $$
```

DIAGRAM TABEL

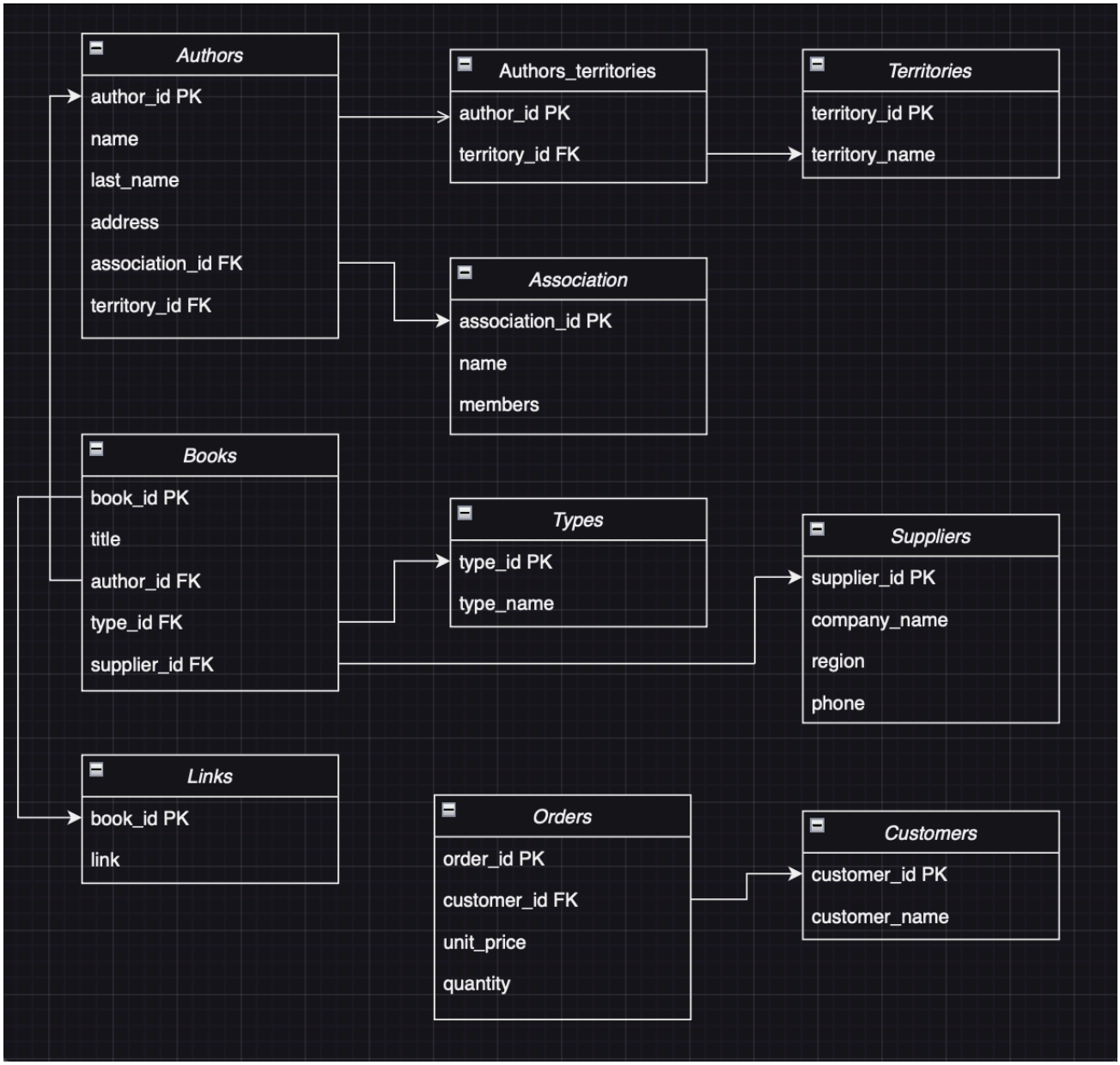
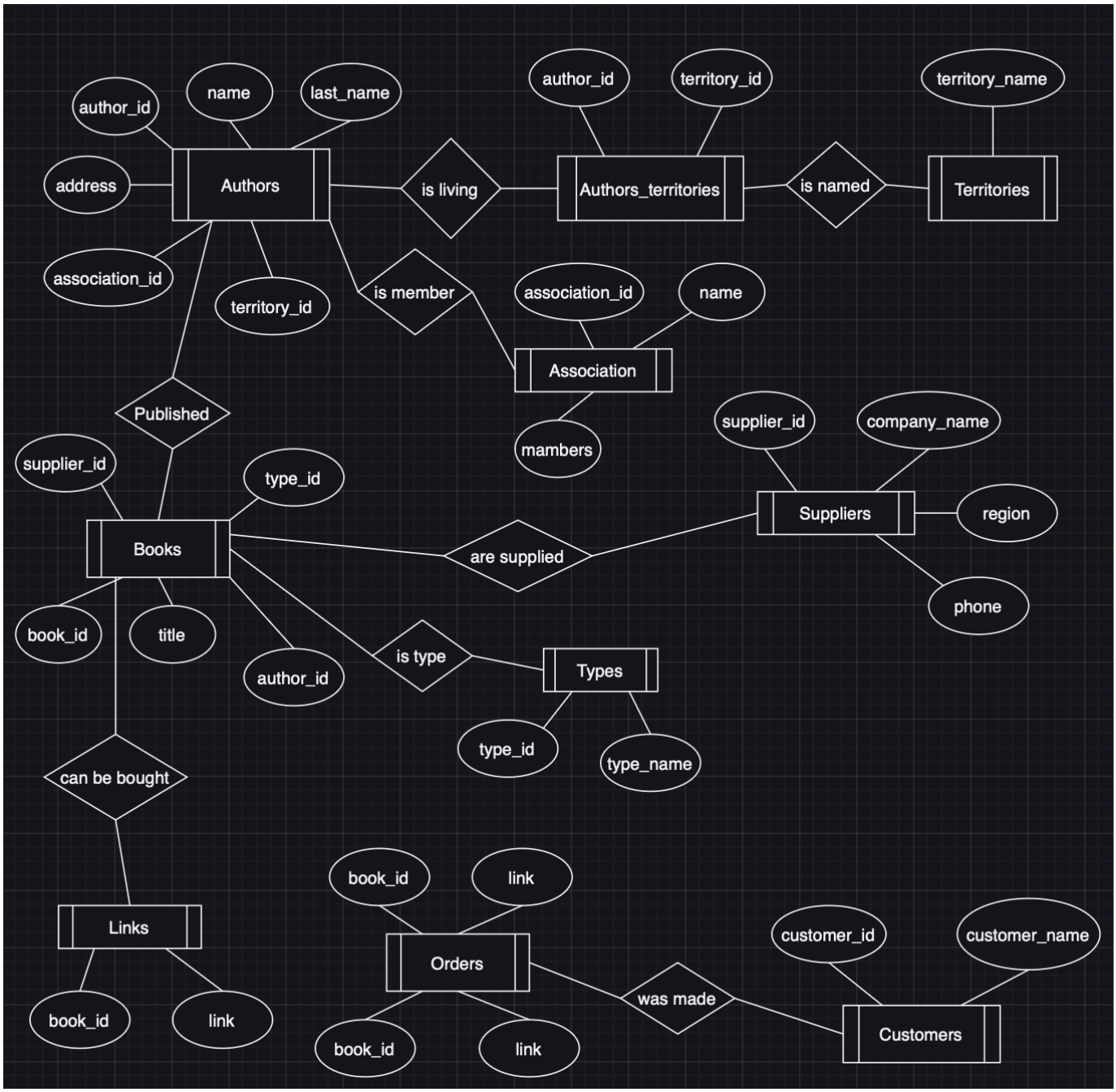


DIAGRAM ER


author_id
name
last_name
Authors
address
association_id
territory_id
is living
author_id
territory_id
Authors_territories
territory_name
is named
Territories
is member
association_id
name
Association
mambers
Published
supplier_id
type_id
Books
book_id
title
author_id
are supplied
supplier_id
company_name
Suppliers
region
phone
is type
Types
type_id
type_name
can be bought
Links
book_id
link
book_id
link
Orders
book_id
link
was made
customer_id
customer_name
Customers

TYPOWE ZAPYTANIA

Typowe zapytania związane są z funkcjami obsługiwanymi przez GUI mojej aplikacji. Są to z pewnością wyszukiwanie książki o danym tytule, zmiana tytułu zaznaczonej książki oraz jej usuwanie. Ponadto często użytkownik będzie chciał ujrzeć zawartość całej biblioteki. Odpowiednia kwerenda zostanie wywołana po wciśnięciu przycisku „View All".

STRATEGIA PIELĘGNACJI BAZY DANYCH

Na początku tworzona jest pełna kopia zapasowa, tworząc na serwerze usługi kopii zapasowej obraz startowy zasobów wskazanych do zabezpieczenia. Na tym etapie użytkownik usługi na dostęp do jednej wersji plików pobranych i zabezpieczonych w chwili utworzenia pełnej kopii zapasowej. Ten rodzaj kopii zajmuje najwięcej miejsca i wykonywany jest najdłużej, zatem będzie stosowany co 2 miesiące.

Co tydzień wykonuje się jednak przyrostowa kopia zapasowa, która bazuje na schemacie tworzenia kopii zapasowej danych, które zostały zmienione lub dodane od czasu utworzenia ostatniej, dowolnej kopii zapasowej. Takie rozwiązanie zajmuje najmniej miejsca w ramach serwera i proces trwa krótko.