|  |  |
| --- | --- |
| **Nazwa przedmiotu** | Bazy Danych |
| **Prowadzący zajęcia** | mgr inż. Piotr Woźniak |
| **Tryb zajęć** | Projekt |
| **Grupa projektowa** | L02 |
| **Imię i nazwisko** | Paweł Torba |

**Spis treści**

[*Określenie i analiza projektu* 3](#__RefHeading___Toc147_2509919536)

[Tematyka i zakres projektu 3](#__RefHeading___Toc731_2509919536)

[Zagadnienia związane z tematem 3](#__RefHeading___Toc733_2509919536)

[Funkcje bazy danych oraz ich priorytety 3](#__RefHeading___Toc735_2509919536)

[Technologia oraz rodzaj bazy danych 4](#__RefHeading___Toc737_2509919536)

[Narzędzia do realizacji projektu 4](#__RefHeading___Toc739_2509919536)

[Repozytorium 4](#__RefHeading___Toc741_2509919536)

[Prezentacja diagramu DB i SQL 5](#__RefHeading___Toc501_3341964130)

[Diagram bazy danych 5](#__RefHeading___Toc503_3341964130)

[Opis tabel bazy danych i ich funkcji 6](#__RefHeading___Toc503_3341964130_Copy_1)

[Przykładowe, proste zapytania SQL 7](#__RefHeading___Toc1088_1385768736)

[Prezentacja zarządzania bazą danych 8](#__RefHeading___Toc1264_2518598345)

[Zaawansowane zapytania SQL 8](#__RefHeading___Toc2143_1316972759)

[Procedury i funkcje obsługujące bazę danych 11](#__RefHeading___Toc2143_1316972759_kopia_)

[Użytkownicy bazy danych i ich role 13](#__RefHeading___Toc1088_1385768736_Copy_2)

[Zarządzanie bazą danych 13](#__RefHeading___Toc1088_1385768736_Copy_4)

# ***Określenie i analiza projektu***

## Tematyka i zakres projektu

Celem projektu jest zaprojektowanie oraz implementacja **systemu zarządzania drzewem genealogicznym rodziny** umożliwiającego:

* gromadzenie i porządkowanie danych o osobach (zarówno żyjących, jak i zmarłych) — dane osobowe, daty, miejsca urodzin i zgonu,
* odwzorowanie relacji rodzinnych (rodzic-dziecko, małżeństwo, partnerstwo, rodzeństwo itp.),
* rejestrowanie zdarzeń (narodziny, ślub, zgon i inne) wraz z czasem i lokalizacją,
* przechowywanie multimediów (zdjęcia/filmy) i łączących je relacji,
* prowadzenie historii zmian oraz wykonywanie kopii zapasowych.

## Zagadnienia związane z tematem

* **Modelowanie więzów pokrewieństwa – obsługa relacji rodzic-dziecko, rodzeństwa, małżeństw, partnerstw oraz więzi adopcyjnych.**
* Zmienność historyczna – rejestracja zmian nazwisk, miejsc zamieszkania.
* Chronologia zdarzeń - tworzenie linii czasu dla wydarzeń związanych z danym drzewem genealogicznym.
* Obsługa zapytań genealogicznych - wyszukiwanie przodków/potomków i określanie stopnia pokrewieństwa.

## Funkcje bazy danych oraz ich priorytety

|  |  |
| --- | --- |
| **Priorytet** | **Funkcja** |
| P1 | Dodawanie, odczyt, aktualizacja i usuwanie podstawowych danych na temat osób, rodzin, miejsc i zdarzeń |
| P1 | Rejestrowanie podstawowych zdarzeń (narodziny, ślub, zgon) |
| P2 | Generowanie drzewa genealogicznego (PDF/PNG) |
| P2 | Zapytania o pokrewieństwo (przodkowie, potomkowie, wspólny przodek) |
| P3 | Import/eksport (CSV) danych dotyczących osób i zdarzeń |
| P3 | Historia zmian i tworzenie kopii zapasowych |
| P3 | Przechowywanie multimediów – przechowywanie wszelkich zdjęć wraz z opisem określającym ich relacje do zdarzeń oraz osób |

Legenda: P1- funkcje podstawowe/krytyczne; P2 – funkcje ważne ale nie krytyczne; P3 – usprawnienia i dodatkowe funkcje

## Technologia oraz rodzaj bazy danych

System zarządzania drzewem genealogicznym (i przechowywaniem wszelkich danych dotyczących zdjęć, zdarzeń i relacji) opierał się będzie na relacyjnej bazie danych ze względu na możliwości jasnego zdefiniowania zależności i spójności referencji pomiędzy kwerendami, elementami bazy danych.

Do realizacji bazy danych wykorzystany zostanie system PostgreSQL, ze względu na zaawansowane typy danych, które uproszczą jej realizację i fakt, że jest to system otwartego oprogramowania („open source”) dostępny na szeroką gammę platform (Linux, Windows, Mac).

## Narzędzia do realizacji projektu

|  |  |
| --- | --- |
| **Obszar** | **Narzędzie** |
| Interfejs SQL | Psql (interfejs wiersza poleceń) |
| Interfejs graficzny systemu zarządzania bazą danych | pgAdmin 4 |
| Kontrola wersji | Git oraz GitHub |
| Język programowania do komunikacji z bazą danych | Python 3.11 (lub wyżej) |
| Edytor kodu źródłowego | Visual Studio Code |

## Repozytorium

URL: *https://github.com/PawelTorba/Geneaologia\_BD/tree/main*

Struktura repozytorium:

Geneaologia\_BD/

├── bd/

│ ├── example\_data

│ ├── functions

├── docs/

├── src/

├── LICENSE

└── README.md

Repozytorium składa się z kilku folderów oraz plików:

* bd/ - folder zawierający podstawowe funkcje SQL odtwarzające bazę danych
* bd/example\_data/ – folder zawierający funkcje SQL uzupełniające bazę danych przykładowymi rekordami
* bd/functions/ - folder zawierający funkcje SQL realizujące przedstawione wcześniej założenia (tworzenie drzew genealogicznych, zapytania o pokrewieństwo, itp.)
* docs/ - folder, w którym znajdzie się dokumentacja bazy danych, z ewentualnymi diagramami
* src/ - folder, w którym znajdą się funkcje z zewnętrznego języka oprogramowania (Python)

# **Prezentacja diagramu DB i SQL**

## Diagram bazy danych

## 

## Opis tabel bazy danych i ich funkcji

|  |  |  |
| --- | --- | --- |
| **Tabela** | **Klucz główny tabeli** | **Funkcja tabeli** |
| Osoby | ID\_Osoba | Przechowuje dane o osobach – imiona, nazwiska, daty urodzin i zgonu, miejscu zamieszkania |
| Związki | ID\_Zwiazek | Przechowuje informacje o małżeństwach/partnerstwach – typ relacji i daty jej trwania |
| Osoby\_Związki | ID\_Osoba\_Zwiazki | Łączy osoby ze związkami, umożliwiając przypisanie wielu osób do jednego związku i odwrotnie |
| Pokrewieństwa | ID\_Pokrewienstwo | Przechowuje stopnie  pokrewieństwa (rodzic-dziecko, brat, adopcja itp.) |
| Osoby\_Pokrewieństwa | ID\_Osoby\_Pokrewienstwo | Zapisuje parę osób wraz z konkretnym stopniem pokrewieństwa |
| Rodziny | ID\_Rodzina | Przechowuje nazwiska rodowe rodzin |
| Osoby\_Rodziny | ID\_Osoba\_Rodzina | Łączy osoby z rodzinami, pozwalając osobie należeć do więcej niż jednej rodziny |
| Zdarzenia | ID\_Zdarzenie | Przechowuje zdarzenia (narodziny, ślub, zgon, przeprowadzka…) wraz z datą i miejscem |
| Osoby\_Zdarzenia | ID\_Osoba\_Zdarzenia | Wiąże osoby ze zdarzeniami i zapisuje ich rolę w zdarzeniu |
| Miejsca | ID\_Miejsce | Przechowuje miejsca geograficzne z nazwą, opisem i współrzędnymi |
| Fotografie | ID\_Zdjecie | Przechowuje pliki zdjęć, ich opis, datę wykonania i miejsce |
| Osoby\_Fotografie | ID\_Osoba\_Fotografie | Łączy osoby ze zdjęciami, wskazując kto znajduje się na fotografii |
| Zdarzenia\_Fotografie | ID\_Zdarzenie\_Fotografie | Łączy multimedia ze zdarzeniami |

## Przykładowe, proste zapytania SQL

|  |  |
| --- | --- |
| **Zapytanie SQL** | **Opis** |
| 1 **INSERT** **INTO** Osoby (Imie, Nazwisko, Data\_Urodzenia) **VALUES** ('Pawel', 'Torba', '2002-06-18'); | Dodaje przykładową osobę do tabeli Osoby |
| 1 **INSERT** **INTO** Rodziny (Nazwisko\_Rodziny) **VALUES** ('Torba'); | Dodaje rodzinę o określonym nazwisku do tabeli Rodziny |
| 1 **INSERT** **INTO** Miejsca (Nazwa\_Miejsca, Opis, Lokalizacja) **VALUES** ('Dukla', 'Miejscowosc', '(42.9383, 11.0614)'); | Dodaje przykładowe miejsce do tabeli Miejsca |
| 1 **INSERT** **INTO** Zdarzenia (ID\_Miejsce, Nazwa\_Zdarzenia, Opis\_Zdarzenia, Data\_Zdarzenia)  2 **VALUES** (  3 (**SELECT** ID\_Miejsce **FROM** Miejsca **WHERE** Nazwa\_Miejsca='Dukla' **LIMIT** 1),  4 'Slub Jana i Anny',  5 'Ceremonia slubna',  6 '2025-01-01'  7 ); | Dodaje przykładowe zdarzenie wraz z przypisanym określonym miejscem |
| 1 **SELECT** Imie, Nazwisko, Data\_Urodzenia **FROM** Osoby **ORDER** **BY** Nazwisko; | Wypisuje określone pola z tabeli Osoby, sortując po nazwisku |
| 1 **SELECT** Zdarzenia.Opis\_Zdarzenia, Zdarzenia.Nazwa\_Zdarzenia **FROM** Zdarzenia **JOIN** Osoby\_Zdarzenia **on** Osoby\_Zdarzenia.ID\_Zdarzenie = Zdarzenia.ID\_Zdarzenie **WHERE** Osoby\_Zdarzenia.ID\_Osoba = 1; | Wypisuje wszystkie zdarzenia związane z konkretną osobą |

# **Prezentacja zarządzania bazą danych**

## Zaawansowane zapytania SQL

|  |  |  |
| --- | --- | --- |
| **Zapytanie SQL (selekcja danych)** | | **Opis** |
| 1 **WITH** **RECURSIVE**  2  3 mapa(stopien, skok) **AS** (  4 **VALUES** ('ojciec',1), ('matka',1),  5 ('dziadek',2), ('babcia',2),  6 ('pradziadek',3), ('prababcia',3)  7 ),  8  9 przodkowie(id\_osoba, pokolenie, sciezka) **AS** (  10 **SELECT**  11 :id\_osoba,  12 0,  13 ARRAY[:id\_osoba]  14  15 **UNION** **ALL**  16  17 **SELECT**  18 op.id\_osoba\_2,  19 p.pokolenie + m.skok,  20 p.sciezka || op.id\_osoba\_2  21 **FROM** przodkowie p  22 **JOIN** osoby\_pokrewienstwa op  23 **ON** op.id\_osoba\_1 = p.id\_osoba  24 **JOIN** pokrewienstwa pk  25 **ON** pk.id\_pokrewienstwo = op.id\_pokrewienstwo  26 **JOIN** mapa m  27 **ON** m.stopien = pk.stopien\_pokrewienstwa  28 **WHERE** **NOT** op.id\_osoba\_2 = **ANY**(p.sciezka)  29 )  30  31 **SELECT** o.id\_osoba,  32 o.imie,  33 o.nazwisko,  34 **MIN**(pokolenie) **AS** pokolenie  35 **FROM** przodkowie p  36 **JOIN** osoby o **ON** o.id\_osoba = p.id\_osoba  37 **GROUP** **BY** o.id\_osoba, o.imie, o.nazwisko  38 **ORDER** **BY** pokolenie, o.nazwisko, o.imie; | | Zapytanie zwraca wszystkich przodków danej osoby, pokolenie po pokoleniu. |
| 1 *-- :p1 – ID pierwszej osoby*  2 *-- :p2 – ID drugiej osoby*  3 **WITH** **RECURSIVE**  4 p1(anc\_id, gen, path) **AS** (  5 *-- SELF (generacja 0)*  6 **SELECT** :p1, 0, ARRAY[5]  7 **UNION** **ALL**  8 *-- RODZICE, DZIADKOWIE, …*  9 **SELECT** op.id\_osoba\_2,  10 p1.gen + 1,  11 path || op.id\_osoba\_2  12 **FROM** p1  13 **JOIN** osoby\_pokrewienstwa op  14 **ON** op.id\_osoba\_1 = p1.anc\_id  15 **AND** op.id\_pokrewienstwo **IN** (5,6)  16 **WHERE** **NOT** op.id\_osoba\_2 = **ANY**(path)  17 ),  18 p2(anc\_id, gen, path) **AS** (  19 **SELECT** :p2, 0, ARRAY[:p2]  20 **UNION** **ALL**  21 **SELECT** op.id\_osoba\_2,  22 p2.gen + 1,  23 path || op.id\_osoba\_2  24 **FROM** p2  25 **JOIN** osoby\_pokrewienstwa op  26 **ON** op.id\_osoba\_1 = p2.anc\_id  27 **AND** op.id\_pokrewienstwo **IN** (5,6)  28 **WHERE** **NOT** op.id\_osoba\_2 = **ANY**(path)  29 )  30 **SELECT** o.id\_osoba,  31 o.imie,  32 o.nazwisko,  33 p1.gen **AS** gen\_p1,  34 p2.gen **AS** gen\_p2  35 **FROM** p1  36 **JOIN** p2 **USING** (anc\_id)  37 **JOIN** osoby o **ON** o.id\_osoba = p1.anc\_id  38 **ORDER** **BY** (p1.gen + p2.gen)  39 **LIMIT** 1; | | Zapytanie znajduje pierwszego wspólnego przodka dla dwóch zadanych osób. |
| 1 **SELECT** z.data\_zdarzenia,  2 z.nazwa\_zdarzenia,  3 z.opis\_zdarzenia,  4 string\_agg(**DISTINCT** o.imie || ' ' || o.nazwisko, ', ') **AS** uczestnicy  5 **FROM** rodziny r  6 **JOIN** osoby\_rodziny orz **ON** orz.id\_rodzina = r.id\_rodzina  7 **JOIN** osoby o **ON** o.id\_osoba = orz.id\_osoba  8 **JOIN** osoby\_zdarzenia oz **ON** oz.id\_osoba = o.id\_osoba  9 **JOIN** zdarzenia z **ON** z.id\_zdarzenie = oz.id\_zdarzenie  10 **WHERE** r.id\_rodzina = :family\_id  11 **GROUP** **BY** z.id\_zdarzenie  12 **ORDER** **BY** z.data\_zdarzenia; | | Układa oś czasu wydarzeń dla zadanej rodziny. |
| **Zapytanie SQL (aktualizacja danych)** | **Opis** | |
| 1 **UPDATE** zwiazki  2 **SET** data\_zakonczenia = '2020-01-01',  3 powod\_zakonczenia = 'rozwód'  4 **WHERE** id\_zwiazek = :id\_zwiazku; | Zapytanie ustawia datę zakończenia związku | |
| 1 **UPDATE** osoby  2 **SET** data\_zgonu = :data\_zgonu  3 **WHERE** id\_osoba = :osoba;  4  5 **WITH** ins **AS** (  6 **INSERT** **INTO** zdarzenia  7 (id\_miejsce, opis\_zdarzenia, nazwa\_zdarzenia, data\_zdarzenia)  8 **VALUES** (:miejsce,  9 'Zgon osoby ID ' || :osoba,  10 'Zgon',  11 :osoba)  12 RETURNING id\_zdarzenie  13 )  14 **INSERT** **INTO** osoby\_zdarzenia (id\_osoba, id\_zdarzenie)  15 **SELECT** :person\_id, id\_zdarzenie **FROM** ins; | Dopisuje do danej osoby datę zgonu oraz tworzy zdarzenie „Zgon” i łączy je z daną osobą. | |
| **Zapytanie SQL (wprowadzanie danych)** | **Opis** | |
| 1 **WITH** np **AS** (  2 **INSERT** **INTO** osoby (imie, nazwisko, data\_urodzenia)  3 **VALUES** (:imie, :nazwisko, :data\_ur) RETURNING id\_osoba),  4 be **AS** (  5 **INSERT** **INTO** zdarzenia (id\_miejsce, opis\_zdarzenia, nazwa\_zdarzenia, data\_zdarzenia)  6 **VALUES** (:miejsce, 'Narodziny '||:imie||' '||:nazwisko,  7 'Narodziny '||:imie||' '||:nazwisko, :data\_ur)  8 RETURNING id\_zdarzenie)  9 **INSERT** **INTO** osoby\_zdarzenia (id\_osoba, id\_zdarzenie)  10 **SELECT** np.id\_osoba, be.id\_zdarzenie **FROM** np, be; | Wprowadza nową osobę i jednocześnie wpisuje zdarzenie związane z jej narodzinami (np.: jeżeli jest to noworodek). | |
| 1 **WITH** z **AS** (  2 **INSERT** **INTO** zwiazki (typ\_relacji, data\_rozpoczecia)  3 **VALUES** ('slub', :**data**) RETURNING id\_zwiazek)  4 **INSERT** **INTO** osoby\_zwiazki (id\_osoba, id\_zwiazek)  5 **SELECT** id, z.id\_zwiazek  6 **FROM** (**VALUES** (:partner1), (:partner2)) **AS** v(id), z; | Zapytanie dodaje nowe małżeństwo i dodaje do niego partnerów | |
| 1 **WITH** nf **AS** (  2 **INSERT** **INTO** rodziny (nazwisko\_rodziny) **VALUES** (:nazwisko) RETURNING id\_rodzina)  3 **INSERT** **INTO** osoby\_rodziny (id\_osoba, id\_rodzina)  4 **SELECT** 16, nf.id\_rodzina **FROM** nf; | Zapytanie dodaje nową rodzinę i zapisuje zadaną osobę jako jej członka. | |

*(za wartości w postaci „:nazwa”, np.: „:osoba\_id” należy wstawić odpowiednią wartość, w zależności od tego jaki rezultat chcemy uzyskać)*

## Procedury i funkcje obsługujące bazę danych

|  |  |
| --- | --- |
| **Kod procedury/funkcji** | **Opis procedury/funkcji** |
| 1 **CREATE** **OR** **REPLACE** **FUNCTION** sp\_dodaj\_osobe (  2 p\_imie TEXT,  3 p\_nazwisko TEXT,  4 p\_drugie\_imie TEXT,  5 p\_data\_urodzenia DATE,  6 p\_data\_zgonu DATE  7 )  8 **RETURNS** INT  9 **LANGUAGE** plpgsql  10 **AS** $$  11 **DECLARE**  12 v\_id INT;  13 **BEGIN**  14 **IF** **EXISTS** (**SELECT** 1 **FROM** Osoby  15 **WHERE** imie=p\_imie **AND** nazwisko=p\_nazwisko  16 **AND** data\_urodzenia=p\_data\_urodzenia) **THEN**  17 RAISE **EXCEPTION** 'Osoba % % ur. % już istnieje',  18 p\_imie, p\_nazwisko, p\_data\_urodzenia;  19 **END** **IF**;  20  21 **INSERT** **INTO** Osoby(imie,nazwisko,drugie\_imie,  22 data\_urodzenia,data\_zgonu)  23 **VALUES** (p\_imie,p\_nazwisko,p\_drugie\_imie,  24 p\_data\_urodzenia,p\_data\_zgonu)  25 RETURNING id\_osoba **INTO** v\_id;  26  27 **RETURN** v\_id;  28 **END**;  29 $$; | Dodaje nową osobę do bazy, sprawdzając, czy taka już nie istnieje. |
| 1 **CREATE** **OR** **REPLACE** **PROCEDURE** sp\_aktualizuj\_osobe (  2 p\_id INT,  3 p\_imie TEXT,  4 p\_nazwisko TEXT,  5 p\_drugie\_imie TEXT ,  6 p\_data\_urodzenia DATE,  7 p\_data\_zgonu DATE  8 )  9 **LANGUAGE** plpgsql  10 **AS** $$  11 **BEGIN**  12 **IF** p\_data\_zgonu **IS** **NOT** **NULL**  13 **AND** p\_data\_zgonu < p\_data\_urodzenia **THEN**  14 RAISE **EXCEPTION** 'Data zgonu < data urodzenia';  15 **END** **IF**;  16  17 **UPDATE** Osoby  18 **SET** imie = p\_imie,  19 nazwisko = p\_nazwisko,  20 drugie\_imie = p\_drugie\_imie,  21 data\_urodzenia = p\_data\_urodzenia,  22 data\_zgonu = p\_data\_zgonu  23 **WHERE** id\_osoba = p\_id;  24  25 **IF** **NOT** **FOUND** **THEN**  26 RAISE **EXCEPTION** 'Nie ma osoby ID=%', p\_id;  27 **END** **IF**;  28 **END**;  29 $$; | Aktualizuje dane osoby, sprawdzając poprawność daty zgonu. |
| 1 **CREATE** **OR** **REPLACE** **PROCEDURE** sp\_usun\_osobe (  2 p\_id INT,  3 p\_force BOOLEAN **DEFAULT** **FALSE**  4 )  5 **LANGUAGE** plpgsql  6 **AS** $$  7 **BEGIN**  8 **IF** **NOT** p\_force **AND** **EXISTS** (  9 **SELECT** 1 **FROM** Osoby\_Pokrewienstwa  10 **WHERE** p\_id **IN** (id\_osoba\_1,id\_osoba\_2)  11 ) **THEN**  12 RAISE **EXCEPTION**  13 'Osoba ID=% ma powiązania genealogiczne – podaj p\_force=TRUE', p\_id;  14 **END** **IF**;  15  16 **DELETE** **FROM** Osoby\_Zwiazki **WHERE** id\_osoba=p\_id;  17 **DELETE** **FROM** Osoby\_Pokrewienstwa **WHERE** id\_osoba\_1=p\_id **OR** id\_osoba\_2=p\_id;  18 **DELETE** **FROM** Osoby\_Zdarzenia **WHERE** id\_osoba=p\_id;  19 **DELETE** **FROM** Osoby\_Fotografie **WHERE** id\_osoba=p\_id;  20 **DELETE** **FROM** Osoby **WHERE** id\_osoba=p\_id;  21  22 **IF** **NOT** **FOUND** **THEN**  23 RAISE **EXCEPTION** 'Nie ma osoby ID=%', p\_id;  24 **END** **IF**;  25 **END**;  26 $$; | Usuwa osobę i wszystkie powiązane dane, chyba że ma relacje rodzinne i p\_force nie jest ustawione. |
| 1 **CREATE** **OR** **REPLACE** **FUNCTION** sp\_dodaj\_zwiazek (  2 p\_osoba1 INT,  3 p\_osoba2 INT,  4 p\_typ\_relacji TEXT **DEFAULT** 'małżeństwo',  5 p\_data\_start DATE **DEFAULT** **CURRENT\_DATE**  6 )  7 **RETURNS** INT  8 **LANGUAGE** plpgsql  9 **AS** $$  10 **DECLARE**  11 v\_id INT;  12 **BEGIN**  13 **IF** **EXISTS** (  14 **SELECT** 1  15 **FROM** Zwiazki z  16 **JOIN** Osoby\_Zwiazki x1 **ON** x1.id\_zwiazek=z.id\_zwiazek  17 **AND** x1.id\_osoba=p\_osoba1  18 **JOIN** Osoby\_Zwiazki x2 **ON** x2.id\_zwiazek=z.id\_zwiazek  19 **AND** x2.id\_osoba=p\_osoba2  20 **WHERE** z.data\_zakonczenia **IS** **NULL**  21 ) **THEN**  22 RAISE **EXCEPTION**  23 'Osoby % i % są już w aktywnym związku', p\_osoba1, p\_osoba2;  24 **END** **IF**;  25  26 **INSERT** **INTO** Zwiazki(typ\_relacji, data\_rozpoczecia)  27 **VALUES** (p\_typ\_relacji, p\_data\_start)  28 RETURNING id\_zwiazek **INTO** v\_id;  29  30 **INSERT** **INTO** Osoby\_Zwiazki(id\_osoba,id\_zwiazek)  31 **VALUES** (p\_osoba1,v\_id),(p\_osoba2,v\_id);  32  33 **RETURN** v\_id;  34 **END**;  35 $$; | Tworzy nowy związek między dwiema osobami, o ile nie istnieje już aktywny. |
| 1 **CREATE** **OR** **REPLACE** **PROCEDURE** sp\_zakoncz\_zwiazek (  2 p\_id\_zwiazek INT,  3 p\_data\_koniec DATE **DEFAULT** **CURRENT\_DATE**,  4 p\_powod TEXT **DEFAULT** **NULL**  5 )  6 **LANGUAGE** plpgsql  7 **AS** $$  8 **BEGIN**  9 **UPDATE** Zwiazki  10 **SET** data\_zakonczenia = p\_data\_koniec,  11 powod\_zakonczenia= p\_powod  12 **WHERE** id\_zwiazek = p\_id\_zwiazek  13 **AND** data\_zakonczenia **IS** **NULL**;  14  15 **IF** **NOT** **FOUND** **THEN**  16 RAISE **EXCEPTION**  17 'Nie znaleziono aktywnego związku ID=%', p\_id\_zwiazek;  18 **END** **IF**;  19 **END**;  20 $$; | Kończy związek, ustawiając datę zakończenia i opcjonalnie powód. |
| 1 **CREATE** **OR** **REPLACE** **PROCEDURE** sp\_dodaj\_pokrewienstwo (  2 p\_os1 INT,  3 p\_os2 INT,  4 p\_id\_pokr INT *-- np. klucz do „rodzic”, „syn” itp.*  5 )  6 **LANGUAGE** plpgsql  7 **AS** $$  8 **BEGIN**  9 **IF** **EXISTS** (  10 **SELECT** 1 **FROM** Osoby\_Pokrewienstwa  11 **WHERE** (id\_osoba\_1,id\_pokrewienstwo,id\_osoba\_2)=(p\_os1,p\_id\_pokr,p\_os2)  12 **OR** (id\_osoba\_1,id\_pokrewienstwo,id\_osoba\_2)=(p\_os2,p\_id\_pokr,p\_os1)  13 ) **THEN**  14 RAISE **EXCEPTION** 'Taka relacja już istnieje.';  15 **END** **IF**;  16  17 **INSERT** **INTO** Osoby\_Pokrewienstwa(id\_osoba\_1,id\_pokrewienstwo,id\_osoba\_2)  18 **VALUES** (p\_os1,p\_id\_pokr,p\_os2);  19 **END**;  20 $$; | Dodaje relację pokrewieństwa między dwiema osobami, jeśli nie istnieje. |
| 1 **CREATE** **OR** **REPLACE** **FUNCTION** sp\_dodaj\_zdarzenie (  2 p\_nazwa TEXT,  3 p\_opis TEXT,  4 p\_data DATE,  5 p\_miejsce INT,  6 p\_uczestnicy INT[] **DEFAULT** **NULL** *-- tablica id\_osoba*  7 )  8 **RETURNS** INT  9 **LANGUAGE** plpgsql  10 **AS** $$  11 **DECLARE**  12 v\_id INT;  13 v\_os INT;  14 **BEGIN**  15 **INSERT** **INTO** Zdarzenia(id\_miejsce,opis\_zdarzenia,  16 nazwa\_zdarzenia,data\_zdarzenia)  17 **VALUES** (p\_miejsce,p\_opis,p\_nazwa,p\_data)  18 RETURNING id\_zdarzenie **INTO** v\_id;  19  20 **IF** p\_uczestnicy **IS** **NOT** **NULL** **THEN**  21 FOREACH v\_os **IN** ARRAY p\_uczestnicy LOOP  22 **INSERT** **INTO** Osoby\_Zdarzenia(id\_osoba,id\_zdarzenie)  23 **VALUES** (v\_os,v\_id);  24 **END** LOOP;  25 **END** **IF**;  26  27 **RETURN** v\_id;  28 **END**;  29 $$; | Tworzy nowe zdarzenie i przypisuje do niego uczestników. |
| 1 **CREATE** **OR** **REPLACE** **PROCEDURE** sp\_merge\_osoby (  2 p\_duplikat INT,  3 p\_oryginal INT  4 )  5 **LANGUAGE** plpgsql  6 **AS** $$  7 **BEGIN**  8 **IF** p\_duplikat = p\_oryginal **THEN**  9 RAISE **EXCEPTION** 'ID duplikatu = ID oryginału – operacja bez sensu.';  10 **END** **IF**;  11  12 *-- klucze w tabelach pośrednich*  13 **UPDATE** osoby\_zwiazki **SET** id\_osoba = p\_oryginal **WHERE** id\_osoba = p\_duplikat;  14 **UPDATE** osoby\_pokrewienstwa **SET** id\_osoba\_1 = p\_oryginal **WHERE** id\_osoba\_1 = p\_duplikat;  15 **UPDATE** osoby\_pokrewienstwa **SET** id\_osoba\_2 = p\_oryginal **WHERE** id\_osoba\_2 = p\_duplikat;  16 **UPDATE** osoby\_zdarzenia **SET** id\_osoba = p\_oryginal **WHERE** id\_osoba = p\_duplikat;  17 **UPDATE** osoby\_fotografie **SET** id\_osoba = p\_oryginal **WHERE** id\_osoba = p\_duplikat;  18 **UPDATE** osoby\_rodziny **SET** id\_osoba = p\_oryginal **WHERE** id\_osoba = p\_duplikat;  19  20 *-- usuń duplikat*  21 **DELETE** **FROM** osoby **WHERE** id\_osoba = p\_duplikat;  22 **END**;  23 $$;  24 **COMMENT** **ON** **PROCEDURE** sp\_merge\_osoby(INT,INT)  25 **IS** 'Scalanie duplikatów osób: przekierowuje FK i usuwa rekord duplikatu.'; | Scala duplikaty osób, przekierowując powiązania i usuwając duplikat. |
| 1 **CREATE** **OR** **REPLACE** **FUNCTION** sp\_waliduj\_relacje ()  2 **RETURNS** **TABLE**(opis TEXT)  3 **LANGUAGE** plpgsql  4 **AS** $$  5 **BEGIN**  6  7 **RETURN** QUERY  8 **SELECT** ' Dziecko starsze od rodzica: dziecko='||**c**.id\_osoba||' ('||**c**.data\_urodzenia||') '  9 ||'rodzic='||p.id\_osoba||' ('||p.data\_urodzenia||')'  10 **FROM** osoby\_pokrewienstwa op  11 **JOIN** pokrewienstwa pk **USING**(id\_pokrewienstwo)  12 **JOIN** osoby p **ON** p.id\_osoba = op.id\_osoba\_1  13 **JOIN** osoby **c** **ON** **c**.id\_osoba = op.id\_osoba\_2  14 **WHERE** pk.stopien\_pokrewienstwa **ILIKE** 'rodzic'  15 **AND** **c**.data\_urodzenia <= p.data\_urodzenia;  16  17  18 **RETURN** QUERY  19 **SELECT** ' Zwiazek po smierci partnera: zw.'||z.id\_zwiazek  20 ||' start='||z.data\_rozpoczecia  21 ||' partner='||o.id\_osoba||' zgon='||o.data\_zgonu  22 **FROM** zwiazki z  23 **JOIN** osoby\_zwiazki oz **USING**(id\_zwiazek)  24 **JOIN** osoby o **ON** o.id\_osoba = oz.id\_osoba  25 **WHERE** o.data\_zgonu **IS** **NOT** **NULL**  26 **AND** z.data\_rozpoczecia > o.data\_zgonu;  27  28 **RETURN** QUERY  29 **SELECT** ' Bigamia: osoba='||id\_osoba||' aktywne zw.'||cnt  30 **FROM** (  31 **SELECT** oz.id\_osoba, **COUNT**(\*) **AS** cnt  32 **FROM** osoby\_zwiazki oz  33 **JOIN** zwiazki z **USING**(id\_zwiazek)  34 **WHERE** z.data\_zakonczenia **IS** **NULL**  35 **GROUP** **BY** oz.id\_osoba  36 ) x  37 **WHERE** cnt > 1;  38  39 **RETURN** QUERY  40 **SELECT** ' Zdarzenie po smierci uczestnika: zd.'||z.id\_zdarzenie  41 ||' ('||z.data\_zdarzenia||') / osoba='||o.id\_osoba  42 ||' zgon='||o.data\_zgonu  43 **FROM** zdarzenia z  44 **JOIN** osoby\_zdarzenia oz **USING**(id\_zdarzenie)  45 **JOIN** osoby o **ON** o.id\_osoba = oz.id\_osoba  46 **WHERE** o.data\_zgonu **IS** **NOT** **NULL**  47 **AND** z.data\_zdarzenia > o.data\_zgonu;  48 **END**;  49 $$;  50 **COMMENT** **ON** **FUNCTION** sp\_waliduj\_relacje()  51 **IS** 'Zwraca listę niespójności (dziecko starsze od rodzica, bigamia, itp.).'; | Wyszukuje logiczne niespójności w danych genealogicznych (np. bigamia, błędne daty). |
| 1 **CREATE** **OR** **REPLACE** **FUNCTION** fn\_licznik\_zdjec (  2 p\_id INT  3 )  4 **RETURNS** INT  5 **LANGUAGE** **SQL** **STABLE**  6 **AS** $$  7 **SELECT** **COUNT**(\*)  8 **FROM** osoby\_fotografie  9 **WHERE** id\_osoba = p\_id;  10 $$;  11 **COMMENT** **ON** **FUNCTION** fn\_licznik\_zdjec(INT)  12 **IS** 'Zwraca liczbe fotografii powiązanych z daną osobą.'; | Zwraca liczbę zdjęć powiązanych z daną osobą. |
| 1 **CREATE** **OR** **REPLACE** **FUNCTION** f\_pokrewienstwa(\_id int)  2 **RETURNS** **TABLE** (  3 imie text,  4 nazwisko text,  5 pokrewienstwo text  6 ) **LANGUAGE** **SQL** **STABLE** **AS**  7 $$  8 **WITH** rel **AS** ( *-- łapiemy wszystkie surowe wiersze*  9 **SELECT**  10 **CASE** **WHEN** op.ID\_Osoba\_1 = \_id  11 **THEN** op.ID\_Osoba\_2 **ELSE** op.ID\_Osoba\_1 **END** **AS** id\_krewnego,  12 op.ID\_Pokrewienstwo **AS** rel\_orig,  13 (op.ID\_Osoba\_1 = \_id) **AS** i\_am\_side1  14 **FROM** Osoby\_Pokrewienstwa op  15 **WHERE** \_id **IN** (op.ID\_Osoba\_1, op.ID\_Osoba\_2)  16 ),  17 mapped **AS** ( *-- zamieniamy relacje na „jak ja je widzę”*  18 **SELECT**  19 o.ID\_Osoba **AS** id\_krewnego,  20 o.Imie,  21 o.Nazwisko,  22 **CASE**  23 **WHEN** i\_am\_side1 **THEN** pk.Stopien\_Pokrewienstwa  24 **ELSE**  25 **CASE** rel\_orig  26 **WHEN** 1 **THEN** **CASE** **WHEN** o.Imie **LIKE** '%a' **THEN** 'prawnuczka' **ELSE** 'prawnuk' **END**  27 **WHEN** 3 **THEN** **CASE** **WHEN** o.Imie **LIKE** '%a' **THEN** 'prawnuczka' **ELSE** 'prawnuk' **END**  28 **WHEN** 2 **THEN** **CASE** **WHEN** o.Imie **LIKE** '%a' **THEN** 'wnuczka' **ELSE** 'wnuk' **END**  29 **WHEN** 4 **THEN** **CASE** **WHEN** o.Imie **LIKE** '%a' **THEN** 'wnuczka' **ELSE** 'wnuk' **END**  30 **WHEN** 5 **THEN** **CASE** **WHEN** o.Imie **LIKE** '%a' **THEN** 'córka' **ELSE** 'syn' **END**  31 **WHEN** 6 **THEN** **CASE** **WHEN** o.Imie **LIKE** '%a' **THEN** 'córka' **ELSE** 'syn' **END**  32 **WHEN** 9 **THEN** **CASE** **WHEN** o.Imie **LIKE** '%a' **THEN** 'matka' **ELSE** 'ojciec' **END**  33 **WHEN** 10 **THEN** **CASE** **WHEN** o.Imie **LIKE** '%a' **THEN** 'matka' **ELSE** 'ojciec' **END**  34 **WHEN** 11 **THEN** **CASE** **WHEN** o.Imie **LIKE** '%a' **THEN** 'siostrzenica' **ELSE** 'siostrzeniec' **END**  35 **WHEN** 12 **THEN** **CASE** **WHEN** o.Imie **LIKE** '%a' **THEN** 'siostrzenica' **ELSE** 'siostrzeniec' **END**  36 **WHEN** 8 **THEN** 'rodzeństwo'  37 **WHEN** 7 **THEN** 'kuzynostwo'  38 **ELSE** pk.Stopien\_Pokrewienstwa  39 **END**  40 **END** **AS** pokrewienstwo  41 **FROM** rel  42 **JOIN** Osoby o **ON** o.ID\_Osoba = rel.id\_krewnego  43 **JOIN** Pokrewienstwa pk **ON** pk.ID\_Pokrewienstwo = rel.rel\_orig  44 )  45 **SELECT** **DISTINCT** **ON** (id\_krewnego, pokrewienstwo) *-- usuwa duplikaty*  46 imie,  47 nazwisko,  48 pokrewienstwo  49 **FROM** mapped  50 **ORDER** **BY** id\_krewnego, pokrewienstwo, nazwisko, imie;  51 $$; | Wyświetla listę krewnych danej osoby z opisem relacji. |
| 1 **CREATE** **OR** **REPLACE** **FUNCTION** sp\_dodaj\_foto (  2 p\_plik BYTEA, *-- plik w BYTEA*  3 p\_opis TEXT,  4 p\_data DATE,  5 p\_miejsce INT **DEFAULT** **NULL**,  6 p\_osoby INT[] **DEFAULT** **NULL**,  7 p\_zdarzenia INT[] **DEFAULT** **NULL**  8 )  9 **RETURNS** INT  10 **LANGUAGE** plpgsql **AS** $$  11 **DECLARE**  12 v\_id INT;  13 v\_x INT;  14 **BEGIN**  15 **INSERT** **INTO** Fotografie(Plik, Opis\_Zdjecia, Data\_Wykonania, ID\_Miejsce)  16 **VALUES** (p\_plik, p\_opis, p\_data, p\_miejsce)  17 RETURNING ID\_Zdjecie **INTO** v\_id;  18  19 **IF** p\_osoby **IS** **NOT** **NULL** **THEN**  20 FOREACH v\_x **IN** ARRAY p\_osoby LOOP  21 **INSERT** **INTO** Osoby\_Fotografie(ID\_Osoba, ID\_Zdjecie)  22 **VALUES** (v\_x, v\_id)  23 **ON** CONFLICT **DO** **NOTHING**;  24 **END** LOOP;  25 **END** **IF**;  26  27 **IF** p\_zdarzenia **IS** **NOT** **NULL** **THEN**  28 FOREACH v\_x **IN** ARRAY p\_zdarzenia LOOP  29 **INSERT** **INTO** Zdarzenia\_Fotografie(ID\_Zdjecie, ID\_Zdarzenie)  30 **VALUES** (v\_id, v\_x)  31 **ON** CONFLICT **DO** **NOTHING**;  32 **END** LOOP;  33 **END** **IF**;  34  35 **RETURN** v\_id;  36 **END**;$$;  37 **COMMENT** **ON** **FUNCTION** sp\_dodaj\_foto(BYTEA,TEXT,DATE,INT,INT[],INT[])  38 **IS** 'Dodaje fotografię i wiąże ją z osobami / zdarzeniami.'; | Dodaje zdjęcie i przypisuje je do osób oraz zdarzeń. |
| 1 **CREATE** **OR** **REPLACE** **FUNCTION** fn\_najblizszy\_zyjacy\_krewny (  2 p\_id INT,  3 p\_max\_poziom INT **DEFAULT** 6  4 )  5 **RETURNS** INT  6 **LANGUAGE** **SQL** **STABLE** **AS** $$  7 **WITH** **RECURSIVE** q **AS** (  8 */\* warstwa 0: start \*/*  9 **SELECT** p\_id **AS** id\_osoba,  10 0 **AS** lvl,  11 ARRAY[p\_id] **AS** path  12 **UNION** **ALL**  13 */\* warstwa kolejna: wszystkie relacje w obie strony \*/*  14 **SELECT** **CASE** **WHEN** op.ID\_Osoba\_1 = q.id\_osoba  15 **THEN** op.ID\_Osoba\_2 **ELSE** op.ID\_Osoba\_1 **END**,  16 q.lvl + 1,  17 path || **CASE** **WHEN** op.ID\_Osoba\_1 = q.id\_osoba  18 **THEN** op.ID\_Osoba\_2 **ELSE** op.ID\_Osoba\_1 **END**  19 **FROM** q  20 **JOIN** Osoby\_Pokrewienstwa op  21 **ON** op.ID\_Osoba\_1 = q.id\_osoba  22 **OR** op.ID\_Osoba\_2 = q.id\_osoba  23 **WHERE** q.lvl < p\_max\_poziom  24 **AND** **NOT** (**CASE** **WHEN** op.ID\_Osoba\_1 = q.id\_osoba  25 **THEN** op.ID\_Osoba\_2 **ELSE** op.ID\_Osoba\_1 **END**) = **ANY**(path)  26 )  27 **SELECT** id\_osoba  28 **FROM** q  29 **JOIN** Osoby o **USING**(id\_osoba)  30 **WHERE** o.Data\_Zgonu **IS** **NULL** *-- żyjący*  31 **AND** id\_osoba <> p\_id *-- pomijamy siebie*  32 **ORDER** **BY** lvl *-- najkrótsza ścieżka*  33 **LIMIT** 1;  34 $$;  35 **COMMENT** **ON** **FUNCTION** fn\_najblizszy\_zyjacy\_krewny(INT,INT)  36 **IS** 'najbliższy żyjący krewny, NULL jeśli brak.'; | Zwraca ID najbliższego żyjącego krewnego danej osoby. |

Plik z procedurami przeznaczony do importu znajduje się w repozytorium, bezpośredni URL: *https://github.com/PawelTorba/Geneaologia\_BD/blob/main/bd/functions/FUNCTIONS\_PROCEDURES.sql*

## Użytkownicy bazy danych i ich role

**GEN\_Administrator:** użytkownik sprawujący pełną kontrolę nad bazą danych w tym strukturą, danymi i kontami użytkowników. Posiada wszystkie uprawnienia (superuser).

**GEN\_Redaktor:** użytkownik zarządzający danymi zawartymi w bazie danych. Może edytować, dodawać oraz usuwać dane. Nie może zarządzać strukturą bazy.

**GEN\_Przegladajacy:** użytkownik mogący jedynie przeglądać dane zawarte w bazie oraz je eksportować. Nie posiada uprawnień umożliwiających edycję zawartości bazy danych.

**Skrypt tworzący użytkowników oraz nadający im uprawnienia:** [https://github.com/PawelTorba/Geneaologia\_BD/blob/main/bd/CREATE\_ACCOUNTS.sql](https://github.com/PawelTorba/Geneaologia_BD/blob/main/bd/CREATE_BACKUP.bat)

## Zarządzanie bazą danych

**Kod pliku wsadowego (.bat) tworzącego punkt przywracania (kopię zapasową) bazy danych:**

<https://github.com/PawelTorba/Geneaologia_BD/blob/main/bd/CREATE_BACKUP.bat>

**Kod pliku wsadowego (.bat) przywracającego bazę danych do określonego punktu przywracania:**

[https://github.com/PawelTorba/Geneaologia\_BD/blob/main/bd/RESTORE\_BACKUP.bat](https://github.com/PawelTorba/Geneaologia_BD/blob/main/bd/CREATE_BACKUP.bat)