|  |  |
| --- | --- |
| **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 'Ślub Jana i Anny',  5 'Ceremonia ślubna',  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 |
| 1 **SELECT** p2.Imie, p2.Nazwisko, pk.Stopien\_Pokrewienstwa **FROM** Osoby\_Pokrewienstwa op **JOIN** Pokrewienstwa pk **ON** pk.ID\_Pokrewienstwo = op.ID\_Pokrewienstwo **JOIN** Osoby p1 **ON** p1.ID\_Osoba = op.ID\_Osoba\_1 **JOIN** Osoby p2 **ON** p2.ID\_Osoba = op.ID\_Osoba\_2 **WHERE** p1.ID\_Osoba = 1; | Wypisuje wszystkie pokrewieństwa dla danej osoby |

# **Prezentacja zarządzania bazą danych**

## Zaawansowane zapytania SQL

|  |  |  |
| --- | --- | --- |
| **Zapytanie SQL (selekcja danych)** | | **Opis** |
| 1 **WITH RECURSIVE** przodkowie **AS** (  2 **SELECT**  3 op.ID\_Osoba\_2 **AS** id\_przodka,  4 1 **AS** pokolenie  5 **FROM** Osoby\_Pokrewienstwa op  6 **JOIN** Pokrewienstwa pk **ON** pk.ID\_Pokrewienstwo = op.ID\_Pokrewienstwo  7 **WHERE** op.ID\_Osoba\_1 = :osoba\_id  8 **AND** pk.Stopien\_Pokrewienstwa = 'rodzic'  9  10 **UNION** **ALL**  11  12 **SELECT**  13 op.ID\_Osoba\_2,  14 p.pokolenie + 1  15 **FROM** przodkowie p  16 **JOIN** Osoby\_Pokrewienstwa op  17 **ON** op.ID\_Osoba\_1 = p.id\_przodka  18 **JOIN** Pokrewienstwa pk  19 **ON** pk.ID\_Pokrewienstwo = op.ID\_Pokrewienstwo  20 **WHERE** pk.Stopien\_Pokrewienstwa = 'rodzic'  21 **AND** (:**g** **IS** **NULL** **OR** p.pokolenie < :**g**)  22 )  23 **SELECT**  24 o.ID\_Osoba,  25 o.Imie,  26 o.Nazwisko,  27 pokolenie  28 **FROM** przodkowie  29 **JOIN** Osoby o **ON** o.ID\_Osoba = id\_przodka  30 ORDER BY pokolenie, Nazwisko, Imie; | | Zapytanie zwraca wszystkich przodków danej osoby, pokolenie po pokoleniu. |
| 1 **WITH** **RECURSIVE** p1 **AS** (  2 **SELECT** op.ID\_Osoba\_2 **AS** id\_przodka  3 **FROM** Osoby\_Pokrewienstwa op  4 **JOIN** Pokrewienstwa pk **ON** pk.ID\_Pokrewienstwo = op.ID\_Pokrewienstwo  5 **WHERE** op.ID\_Osoba\_1 = :osoba1 **AND** pk.Stopien\_Pokrewienstwa = 'rodzic'  6 **UNION**  7 **SELECT** op.ID\_Osoba\_2 **FROM** Osoby\_Pokrewienstwa op  8 **JOIN** p1 **ON** op.ID\_Osoba\_1 = p1.id\_przodka  9 **JOIN** Pokrewienstwa pk **ON** pk.ID\_Pokrewienstwo = op.ID\_Pokrewienstwo  10 **WHERE** pk.Stopien\_Pokrewienstwa = 'rodzic'  11 ),  12 p2 **AS** (  13 **SELECT** op.ID\_Osoba\_2 **AS** id\_przodka  14 **FROM** Osoby\_Pokrewienstwa op  15 **JOIN** Pokrewienstwa pk **ON** pk.ID\_Pokrewienstwo = op.ID\_Pokrewienstwo  16 **WHERE** op.ID\_Osoba\_1 = :osoba2 **AND** pk.Stopien\_Pokrewienstwa = 'rodzic'  17 **UNION**  18 **SELECT** op.ID\_Osoba\_2 **FROM** Osoby\_Pokrewienstwa op  19 **JOIN** p2 **ON** op.ID\_Osoba\_1 = p2.id\_przodka  20 **JOIN** Pokrewienstwa pk **ON** pk.ID\_Pokrewienstwo = op.ID\_Pokrewienstwo  21 **WHERE** pk.Stopien\_Pokrewienstwa = 'rodzic'  22 )  23 **SELECT** o.\*  24 **FROM** Osoby o  25 **WHERE** o.ID\_Osoba **IN** (**SELECT** id\_przodka **FROM** p1)  26 **AND** o.ID\_Osoba **IN** (**SELECT** id\_przodka **FROM** p2)  27 **ORDER** **BY** (**SELECT** **COUNT**(\*) **FROM** p1 **WHERE** id\_przodka = o.ID\_Osoba) +  28 (**SELECT** **COUNT**(\*) **FROM** p2 **WHERE** id\_przodka = o.ID\_Osoba)  29 **LIMIT** 1; | | Zapytanie znajduje pierwszego wspólnego przodka dla dwóch zadanych osób. |
| 1 **SELECT**  2 z.Data\_Zdarzenia,  3 z.Nazwa\_Zdarzenia,  4 z.Opis\_Zdarzenia,  5 STRING\_AGG(o.Imie || ' ' || o.Nazwisko, ', ') **AS** Uczestnicy  6 **FROM** Rodziny r  7 **JOIN** Osoby\_Rodziny orz **ON** orz.ID\_Rodzina = r.ID\_Rodzina  8 **JOIN** Osoby\_Zdarzenia oz **ON** oz.ID\_Osoba = orz.ID\_Osoba  9 **JOIN** Zdarzenia z **ON** z.ID\_Zdarzenie = oz.ID\_Zdarzenie  10 **JOIN** Osoby o **ON** o.ID\_Osoba = orz.ID\_Osoba  11 **WHERE** r.ID\_Rodzina = :rodzina\_id  12 **GROUP** **BY** z.ID\_Zdarzenie | | Układa oś czasu wydarzeń dla zadanej rodziny. |
| **Zapytanie SQL (aktualizacja danych)** | **Opis** | |
| 1 **UPDATE** Zwiazki  2 **SET** Data\_Zakonczenia = :data\_rozwodu  3 **WHERE** ID\_Zwiazek = :zwiazek\_id;  4  5 **UPDATE** Osoby\_Zwiazki  6 **SET** Rola = 'były\_' || Rola  7 **WHERE** ID\_Zwiazek = :zwiazek\_id; | Zapytanie ustawia datę zakończenia związku i oznacza rolę partnera/partnerki na „były”. | |
| **UPDATE** Osoby  **SET** Data\_Zgonu = :data\_zgonu  **WHERE** ID\_Osoba = :osoba\_id;  **INSERT** **INTO** Zdarzenia (ID\_Miejsce, Nazwa\_Zdarzenia, Opis\_Zdarzenia, Data\_Zdarzenia)  **VALUES** (:miejsce\_id, 'Zgon', 'Rejestracja zgonu', :data\_zgonu)  RETURNING ID\_Zdarzenie **INTO** :zgon\_event\_id;  **INSERT** **INTO** Osoby\_Zdarzenia (ID\_Osoba, ID\_Zdarzenie, Rola)  **VALUES** (:osoba\_id, :zgon\_event\_id, 'zmarły'); | Dopisuje do danej osoby datę zgonu oraz tworzy zdarzenie „Zgon” i łączy je z daną osobą. | |
| **Zapytanie SQL (wprowadzanie danych)** | **Opis** | |
| 1 **WITH** nowa\_osoba **AS** (  2 **INSERT** **INTO** Osoby (Imie, Nazwisko, Data\_Urodzenia, ID\_Miejsce\_Urodzenia)  3 **VALUES** (:imie, :nazwisko, :data\_urodzenia, :miejsce\_urodzenia\_id)  4 RETURNING ID\_Osoba  5 ),  6 zdarzenie\_narodziny **AS** (  7 **INSERT** **INTO** Zdarzenia (ID\_Miejsce, Nazwa\_Zdarzenia, Data\_Zdarzenia, Opis\_Zdarzenia)  8 **VALUES** (:miejsce\_urodzenia\_id, 'Narodziny', :data\_urodzenia, 'Narodziny osoby')  9 RETURNING ID\_Zdarzenie  10 )  11 **INSERT** **INTO** Osoby\_Zdarzenia (ID\_Osoba, ID\_Zdarzenie, Rola)  12 **SELECT** **no**.ID\_Osoba, zn.ID\_Zdarzenie, 'noworodek'  13 **FROM** nowa\_osoba **no**, zdarzenie\_narodziny zn; | Wprowadza nową osobę i jednocześnie wpisuje zdarzenie związane z jej narodzinami (np.: jeżeli jest to noworodek). | |
| 1 **WITH** malzenstwo **AS** (  2 **INSERT** **INTO** Zwiazki (Typ\_Zwiazku, Data\_Rozpoczecia)  3 **VALUES** ('małżeństwo', :data\_slubu)  4 RETURNING ID\_Zwiazek  5 )  6 **INSERT** **INTO** Osoby\_Zwiazki (ID\_Zwiazek, ID\_Osoba, Rola)  7 **VALUES**  8 ((**SELECT** ID\_Zwiazek **FROM** malzenstwo), :maz\_id, 'mąż'),  9 ((**SELECT** ID\_Zwiazek **FROM** malzenstwo), :zona\_id, 'żona'); | Zapytanie dodaje nowe małżeństwo i dodaje do niego partnerów | |
| 1 **WITH** nowa\_rodzina **AS** (  2 **INSERT** **INTO** Rodziny (Nazwisko\_Rodziny)  3 **VALUES** (:nazwisko\_rodzinne)  4 RETURNING ID\_Rodzina  5 )  6 **INSERT** **INTO** Osoby\_Rodziny (ID\_Osoba, ID\_Rodzina, Data\_Od)  7 **SELECT** :osoba\_id, ID\_Rodzina, **CURRENT\_DATE** **FROM** nowa\_rodzina; | 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** |
| **CREATE** **OR** **REPLACE** **FUNCTION** stopien\_pokrewienstwa(os1 int, os2 int)  **RETURNS** int  **LANGUAGE** **sql**  **AS** $$  **WITH** **RECURSIVE** rel(lvl, id\_curr, path) **AS** (  **SELECT** 0,  os1,  ARRAY[os1]  **UNION** **ALL**  **SELECT** lvl + 1,  **CASE** **WHEN** op.id\_osoba\_1 = id\_curr  **THEN** op.id\_osoba\_2  **ELSE** op.id\_osoba\_1  **END** **AS** next\_id,  path || **CASE** **WHEN** op.id\_osoba\_1 = id\_curr  **THEN** op.id\_osoba\_2  **ELSE** op.id\_osoba\_1  **END** **AS** new\_path  **FROM** rel  **JOIN** osoby\_pokrewienstwa op  **ON** op.id\_osoba\_1 = id\_curr **OR** op.id\_osoba\_2 = id\_curr  **WHERE** **NOT** (  **CASE** **WHEN** op.id\_osoba\_1 = id\_curr  **THEN** op.id\_osoba\_2  **ELSE** op.id\_osoba\_1  **END** = **ANY**(path)  )  )  **SELECT** lvl  **FROM** rel  **WHERE** id\_curr = os2  **ORDER** **BY** lvl  **LIMIT** 1;  $$; | Funkcja zwraca stopień pokrewieństwa dla dwóch zadanych osób. |
| **CREATE** **OR** **REPLACE** **FUNCTION** genealogy\_tree(  p\_person\_id int,  p\_depth int **DEFAULT** **NULL**  )  **RETURNS** **TABLE** (  **level** int,  person\_id int,  first\_name text,  last\_name text  )  **LANGUAGE** **sql**  **STABLE**  **AS** $$  **WITH** **RECURSIVE** anc(**level**, id) **AS** (  *-- poziom 0: osoba wyjsciowa*  **SELECT** 0, p\_person\_id  **UNION** **ALL**  *-- kolejne poziomy: tylko relacja rodzic -> dziecko*  **SELECT** **level** + 1,  op.id\_osoba\_2  **FROM** anc  **JOIN** osoby\_pokrewienstwa op **ON** op.id\_osoba\_1 = anc.id  **JOIN** pokrewienstwa pk **ON** pk.id\_pokrewienstwo = op.id\_pokrewienstwo  **WHERE** pk.stopien\_pokrewienstwa = 'rodzic'  **AND** (p\_depth **IS** **NULL** **OR** **level** < p\_depth)  )  **SELECT**  **level**,  o.id\_osoba,  o.imie,  o.nazwisko  **FROM** anc  **JOIN** osoby o **ON** o.id\_osoba = anc.id  **ORDER** **BY** **level**, o.nazwisko, o.imie;  $$; | Funkcja generuje drzewo genealogiczne dla zadanej osoby z zadaną „głębokością”. |
| **CREATE** **OR** **REPLACE** **PROCEDURE** dodaj\_osobe\_z\_narodzinami(  p\_imie text,  p\_nazwisko text,  p\_data\_ur date,  p\_miejsce\_ur int,  **OUT** new\_person\_id int)  **LANGUAGE** plpgsql  **AS** $$  **DECLARE**  v\_event\_id int;  **BEGIN**  **INSERT** **INTO** osoby (imie, nazwisko, data\_urodzenia, id\_miejsce\_urodzenia)  **VALUES** (p\_imie, p\_nazwisko, p\_data\_ur, p\_miejsce\_ur)  RETURNING id\_osoba **INTO** new\_person\_id;  **INSERT** **INTO** zdarzenia (id\_miejsce, nazwa\_zdarzenia, data\_zdarzenia, opis\_zdarzenia)  **VALUES** (p\_miejsce\_ur, 'Narodziny', p\_data\_ur, 'Narodziny ' || p\_imie)  RETURNING id\_zdarzenie **INTO** v\_event\_id;  **INSERT** **INTO** osoby\_zdarzenia (id\_osoba, id\_zdarzenie, rola)  **VALUES** (new\_person\_id, v\_event\_id, 'noworodek');  **END** $$; | Procedura dodaje osobę do bazy danych oraz tworzy wydarzenie jej narodzin. |
| **CREATE** **OR** **REPLACE** **PROCEDURE** zarejestruj\_malzenstwo(  p\_maz\_id int,  p\_zona\_id int,  p\_data\_slubu date)  **LANGUAGE** plpgsql  **AS** $$  **DECLARE**  v\_z int;  **BEGIN**  **INSERT** **INTO** zwiazki (typ\_zwiazku, data\_rozpoczecia)  **VALUES** ('malzenstwo', p\_data\_slubu)  RETURNING id\_zwiazek **INTO** v\_z;  **INSERT** **INTO** osoby\_zwiazki (id\_zwiazek, id\_osoba, rola)  **VALUES** (v\_z, p\_maz\_id, 'maz'),  (v\_z, p\_zona\_id, 'zona');  **END** $$; | Procedura dodaje małżeństwo do bazy danych wraz z datą rozpoczęcia związku. |

## 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)