# MatrimonialBureauDB

**Miraslau Alkhovik 248655**
**Praskouya Horbach 248656**

## 1 . Podstawowe założenia projektu

**Cel projektu:** stworzenie obiektowo-relacyjnej bazy danych dla biura matrymonialnego, umożliwiającej zarządzanie klientami, ich preferencjami, przypisaniem do opiekunów, organizowaniem randek oraz monitorowaniem działań za pomocą logowania operacji. System ma wspierać funkcjonalności związane z dodawaniem, edytowaniem i przeglądaniem danych w sposób intuicyjny i zgodny z założeniami obiektowego modelowania danych.

**Główne założenia:**

1. **Modelowanie obiektowe:** Wykorzystanie obiektowych typów danych w Oracle (np. `TypAdres`, `TypOsoba`, `TypKlient`).
2. **Integracja logiki biznesowej:**
    - Pakiety i procedury (`PakietBiuro`) obsługujące dodawanie, aktualizację oraz pobieranie danych.
    - Zastosowanie funkcji i procedur w typach obiektowych, np. `pokaz_dane`, `opis`.
3. **Logowanie operacji:** Tabela `LogOperacji` rejestruje wszystkie zmiany wprowadzane w danych klientów.
4. **Walidacja danych oraz biezpieczeństwo:** Triggery sprawdzające poprawność danych (Klient nie może mieć daty urodzenia w przyszłości, Klienci nie mogą

umawiać się na randki z samymi sobą, etc. )

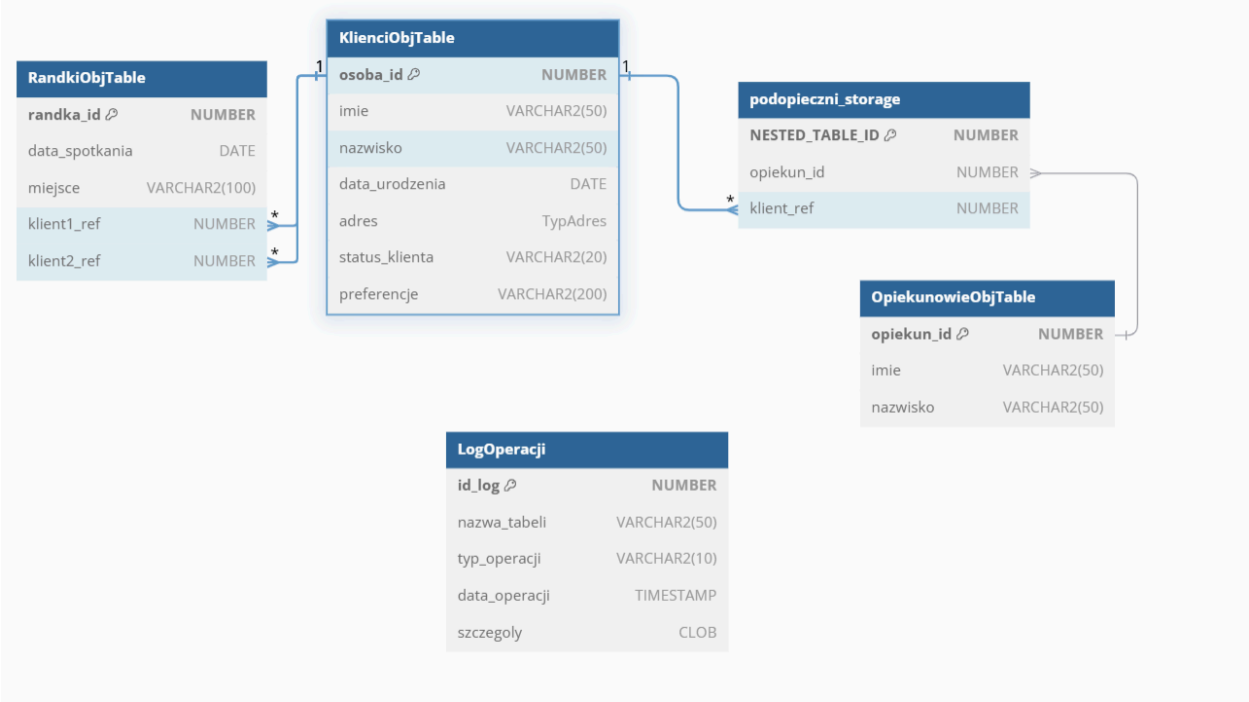
## Możliwości:

1. **Dodawanie danych:** Możliwość dodawania nowych klientów, opiekunów oraz randek + Automatyczna walidacja i przetwarzanie danych w oparciu o obiektowe metody i procedury.
2. **Relacje i referencje:** Obsługa złożonych relacji między obiektami (klient-opiekun, klient-randka).
3. **Wyświetlanie danych:** Funkcje i procedury umożliwiające pobieranie danych o klientach, opiekunach i randkach w ustrukturyzowanej formie.
4. **Logowanie operacji:**Śledzenie historii zmian w tabeli klientów, umożliwiające audyt działań.

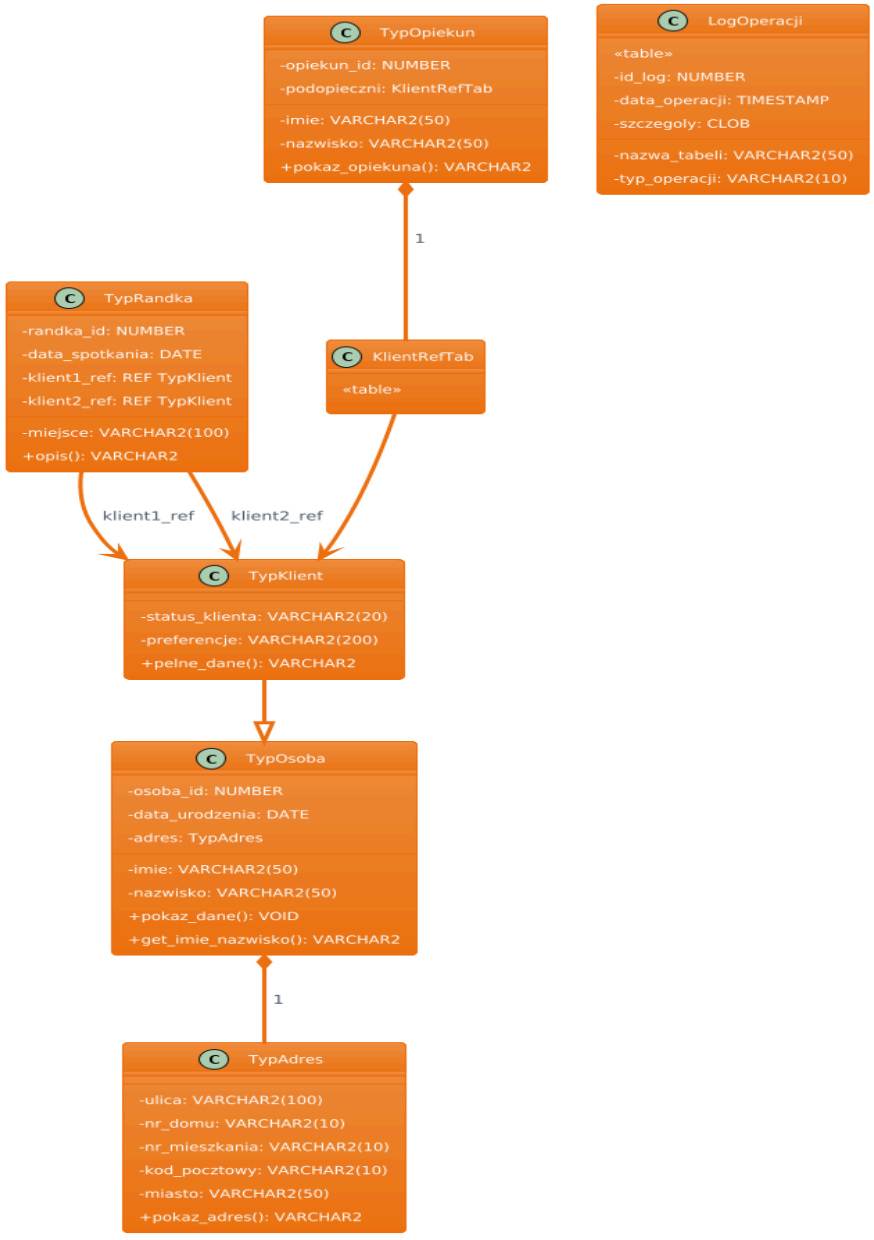
## Ograniczenia przyjęte przy projektowaniu:

1. **Wymóg zgodności z Oracle Database:** System wykorzystuje typy obiektowe i mechanizmy specyficzne dla Oracle, co ogranicza przenośność na inne systemy bazodanowe.
2. **Brak złożonych mechanizmów raportowania:** System koncentruje się na podstawowych funkcjach CRUD, logowaniu i walidacji, bez zaawansowanego raportowania.
3. **Brak automatycznego usuwania nieużywanych obiektów:** Referencje (REF) nie są automatycznie usuwane, co wymaga ręcznego zarządzania relacjami przy usuwaniu klientów.

# 2 . Schemat bazy danych



# 3 . Diagram relacji obiektow



# 4. Obiekty

**1. Obiekty typów danych**

1. **TypAdres**
    - Obiekt opisujący adres, z polami przechowującymi szczegóły takie jak ulica, numer domu, numer mieszkania, kod pocztowy i miasto.
    - **Pola:**
        - `ulica`: VARCHAR2(100) – Nazwa ulicy.
        - `nr_domu`: VARCHAR2(10) – Numer domu.
        - `nr_mieszkania`: VARCHAR2(10) – Numer mieszkania (opcjonalne).
        - `kod_pocztowy`: VARCHAR2(10) – Kod pocztowy.
        - `miasto`: VARCHAR2(50) – Miasto.
    - **Metody:**
        - `pokaz_adres`: Funkcja zwracająca pełny adres w formie tekstowej.
2. **TypOsoba**
    - Obiekt bazowy opisujący ogólne dane osoby.
    - **Pola:**
        - `osoba_id`: NUMBER – Identyfikator osoby (klucz główny).
        - `imie`: VARCHAR2(50) – Imię osoby.
        - `nazwisko`: VARCHAR2(50) – Nazwisko osoby.
        - `data_urodzenia`: DATE – Data urodzenia.
        - `adres`: TypAdres – Obiekt adresu przypisany do osoby.
    - **Metody:**
        - `pokaz_dane`: Procedura wypisująca dane osoby w konsoli.
        - `get_imie_nazwisko`: Funkcja zwracająca pełne imię i nazwisko.
3. **TypKlient**
    - Obiekt dziedziczący po `TypOsoba`, opisujący klientów biura matrymonialnego.
    - **Pola:**
        - `status_klienta`: VARCHAR2(20) – Status klienta (np. "aktywny", "premium").
        - `preferencje`: VARCHAR2(200) – Preferencje klienta (np. "kino,

spacery").
    - **Metody:**
        - `pelne_dane`: Funkcja zwracająca pełne dane klienta w formie tekstowej.
4. **TypRandka**
    - Obiekt opisujący randkę.
    - **Pola:**
        - `randka_id`: NUMBER – Identyfikator randki (klucz główny).
        - `data_spotkania`: DATE – Data randki.
        - `miejsce`: VARCHAR2(100) – Miejsce randki.
        - `klient1_ref`: REF TypKlient – Referencja do pierwszego uczestnika randki.
        - `klient2_ref`: REF TypKlient – Referencja do drugiego uczestnika randki.
    - **Metody:**
        - `opis`: Funkcja zwracająca szczegóły randki.
5. **TypOpiekun**
    - Obiekt opisujący opiekuna zarządzającego klientami.
    - **Pola:**
        - `opiekun_id`: NUMBER – Identyfikator opiekuna (klucz główny).
        - `imie`: VARCHAR2(50) – Imię opiekuna.
        - `nazwisko`: VARCHAR2(50) – Nazwisko opiekuna.
        - `podopieczni`: KlientRefTab – Zagnieżdżona tabela przechowująca referencje do klientów.
    - **Metody:**
        - `pokaz_opiekuna`: Funkcja zwracająca dane opiekuna.
6. **KlientRefTab**
    - Typ tabeli zagnieżdżonej, przechowujący referencje (`REF TypKlient`) do obiektów `TypKlient`.

---

**2. Tabele bazy danych**

1. **KlienciObjTable**

- Tabela przechowująca dane klientów (`TypKlient`).
- **Kolumny:**
  - `osoba_id`: NUMBER – Klucz główny.
  - `imie`, `nazwisko`, `data_urodzenia`, `adres`, `status_klienta`, `preferencje`.
- **Ograniczenia:**
  - `PRIMARY KEY (osoba_id)`.

2. **RandkiObjTable**

   - Tabela przechowująca dane randek (`TypRandka`).
   - **Kolumny:**
     - `randka_id`: NUMBER – Klucz główny.
     - `data_spotkania`, `miejsce`, `klient1_ref`, `klient2_ref`.
   - **Ograniczenia:**
     - `PRIMARY KEY (randka_id)`.

3. **OpiekunowieObjTable**

   - Tabela przechowująca dane opiekunów (`TypOpiekun`).
   - **Kolumny:**
     - `opiekun_id`: NUMBER – Klucz główny.
     - `imie`, `nazwisko`, `podopieczni`.
   - **Ograniczenia:**
     - `PRIMARY KEY (opiekun_id)`.

4. **podopieczni_storage**

   - Tabela zagnieżdżona wspierająca dane o przypisanych klientach (`KlientRefTab`).
   - **Kolumny:**
     - `NESTED_TABLE_ID`: NUMBER – Identyfikator tabeli zagnieżdżonej.
     - `klient_ref`: REF TypKlient – Referencja do klienta.
   - **Ograniczenia:**
     - `PRIMARY KEY (NESTED_TABLE_ID)`.

5. **LogOperacji**

   - Tabela przechowująca historię operacji na tabeli `KlienciObjTable`.
   - **Kolumny:**

        - `id_log`: NUMBER – Klucz główny.
        - `nazwa_tabeli`: VARCHAR2(50) – Nazwa tabeli, której dotyczy operacja.
        - `typ_operacji`: VARCHAR2(10) – Typ operacji (INSERT, UPDATE, DELETE).
        - `data_operacji`: TIMESTAMP – Data operacji.
        - `szczegoly`: CLOB – Szczegóły operacji.

---

**3. Pakiety i triggery**

1. **Pakiet `PakietBiuro`**

    - Zawiera procedury i funkcje do obsługi klientów, opiekunów i randek.
    - **Przykładowe procedury:**
        - `dodaj_klienta`: Dodaje nowego klienta do `KlienciObjTable`.
        - `wyswietl_klienta`: Wyświetla dane konkretnego klienta.
        - `dodaj_randke`: Dodaje nową randkę do `RandkiObjTable`.
        - `dodaj_opiekuna`: Dodaje opiekuna do `OpiekunowieObjTable`.
2. **Triggery:**

    - `trg_check_randka_self`: Sprawdza, czy klient nie jest przypisany jako oba uczestnicy tej samej randki.
    - `trg_check_data_urodzenia`: Weryfikuje, czy data urodzenia klienta nie jest w przyszłości.
    - `trg_log_operacji`: Rejestruje operacje na tabeli `KlienciObjTable` w `LogOperacji`.

# 5 . Role i uprawnienia

1. **`Administrator`:**
    - Ma pełne uprawnienia do zarządzania bazą, w tym:
        - Dostęp do wszystkich tabel, typów i procedur.
        - Możliwość modyfikowania struktury bazy (tworzenie, usuwanie tabel i użytkowników).
        - Wywoływanie procedur i funkcji z pakietu `PakietBiuro`.
    - Uprawnienia zapewniają pełną kontrolę nad systemem.

2. **`Employee`:**
    - Ma ograniczone uprawnienia, dostosowane do codziennej pracy z danymi:
        - Może dodawać i aktualizować dane klientów oraz randek.
        - Może przeglądać dane klientów, randek i opiekunów.
        - Może wywoływać procedury z pakietu `PakietBiuro`.
    - Nie ma możliwości usuwania danych ani modyfikowania struktury bazy.