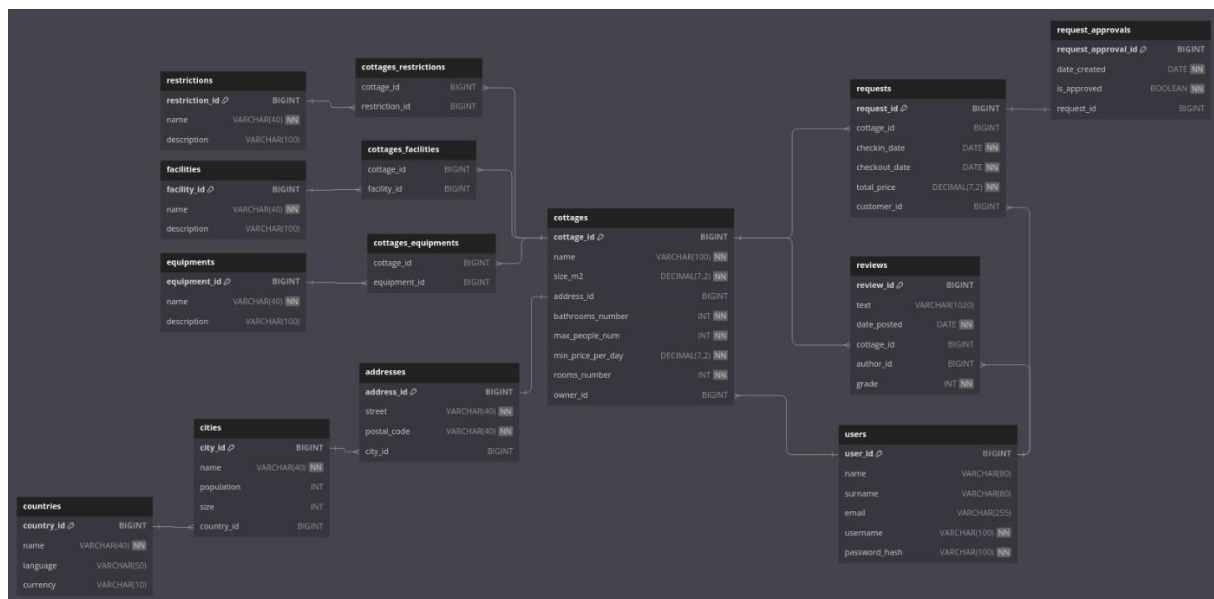


# BD1 – Opis rozwiązania

## Aplikacja umożliwia:

- Rejestrację i logowanie
- Dodawanie, usuwanie i edytowanie domów
- Wyszukiwanie domów
- Dodawanie i usuwanie rezerwacji
- Zatwierdzanie rezerwacji
- Dodawanie i usuwanie opinii

## Model ER



## Tabele:

### 1. Users

**Opis:** Przechowuje informacje o użytkownikach: imię, nazwisko, adres e-mail, nazwę użytkownika i zaszyfrowane hasło.

### Relacje:

- Użytkownik może być właścicielem wielu domków.
- Użytkownik może tworzyć wiele opinii.

- Użytkownik może składać wiele rezerwacji.

## 2. Cottages

**Opis:** Przechowuje szczegółowe informacje o domach: nazwę, wielkość (m<sup>2</sup>), adres, liczbę pokoi (łazienek), maks. liczbę osób, cenę za dzień i właściciela.

**Relacje:**

- Każdy domek ma właściciela (użytkownik).
- Każdy domek ma przypisany adres.
- Każdy domek może mieć wiele rezerwacji, opinii, udogodnień, wyposażenia i ograniczeń.

## 3. Addresses

**Opis:** Przechowuje adresy przypisane do domów: ulicę, kod pocztowy i miasto.

**Relacje:**

- Każdy adres jest powiązany z miastem.
- Każdy domek ma przypisany jeden adres.

## 4. Cities

**Opis:** Przechowuje informacje o miastach: nazwę, populację, wielkość i kraj.

**Relacje:**

- Każde miasto jest powiązane z krajem.
- Każde miasto może mieć wiele adresów.

## 5. Countries

**Opis:** Przechowuje informacje o krajach: nazwę, język urzędowy i walutę.

**Relacje:**

- Każdy kraj może mieć wiele miast.

## 6. Facilities

**Opis:** Przechowuje informacje o dostępnych udogodnieniach: nazwę i opis.

## 7. Cottages\_facilities

**Opis:** Łączy domki z udogodnieniami. Dany dom może mieć wiele udogodnień.

## 8. Equipments

**Opis:** Przechowuje informacje o wyposażeniu: nazwę i opis.

## 9. Cottages\_equipments

**Opis:** Łączy domy z wyposażeniem. Dany dom może mieć wiele elementów wyposażenia.

## 10. Restrictions

**Opis:** Przechowuje informacje o ograniczeniach: nazwę i opis.

## 11. Cottages\_restrictions

**Opis:** Łączy domki z ograniczeniami. Dany dom może mieć wiele ograniczeń.

## 12. Requests

**Opis:** Przechowuje informacje o rezerwacjach: którego domu dotyczy, datę zameldowania i wymeldowania, całkowitą cenę i danego klienta.

## 13. Request\_approvals

**Opis:** Przechowuje informacje o zatwierdzeniach rezerwacji: datę utworzenia zatwierdzenia, status (zatwierdzony lub odrzucony) i rezerwację, której dotyczy.

## 14. Reviews

**Opis:** Przechowuje opinie użytkowników: treść, datę publikacji, ocenę, dom, którego dotyczy i użytkownika wystawiającego opinię.

## Sekwencje

Wszystkie sekwencje mają podobną strukturę:

```
CREATE SEQUENCE COUNTRIES_SEQ START WITH 1 INCREMENT BY 1;
CREATE SEQUENCE CITIES_SEQ START WITH 1 INCREMENT BY 1;
CREATE SEQUENCE ADDRESSES_SEQ START WITH 1 INCREMENT BY 1;
CREATE SEQUENCE USERS_SEQ START WITH 1 INCREMENT BY 1;
CREATE SEQUENCE COTTAGES_SEQ START WITH 1 INCREMENT BY 1;
CREATE SEQUENCE FACILITIES_SEQ START WITH 1 INCREMENT BY 1;
CREATE SEQUENCE EQUIPMENTS_SEQ START WITH 1 INCREMENT BY 1;
CREATE SEQUENCE RESTRICTIONS_SEQ START WITH 1 INCREMENT BY 1;
CREATE SEQUENCE REVIEWS_SEQ START WITH 1 INCREMENT BY 1;
CREATE SEQUENCE REQUESTS_SEQ START WITH 1 INCREMENT BY 1;
CREATE SEQUENCE REQUEST_APPROVALS_SEQ START WITH 1 INCREMENT BY 1;
```

- Zaczynają się od 1.
- Inkrementują o 1.
- Służą do generowania unikalnych identyfikatorów dla odpowiednich tabel.

## Funkcje

1. **is\_grade\_good(v\_grade number)** – Zwraca 1, jeśli ocena jest większa od 3, w przeciwnym razie 0.

```
CREATE OR REPLACE FUNCTION IS_GRADE_GOOD(V_GRADE NUMBER)
RETURN NUMBER
AS
    V_RESULT NUMBER;
BEGIN
    IF V_GRADE > 3 THEN
        V_RESULT := 1;
    ELSE
        V_RESULT := 0;
    END IF;
    RETURN V_RESULT;
END;
```

**Przykład:** is\_grade\_good(2) – 0, is\_grade\_good(4) – 1

2. **request\_status(p\_request\_id number)** – Sprawdza status rezerwacji, zwraca:
  - Accepted – Rezerwacja została rozpatrzona pozytywnie (istnieje request\_approval o is\_approved = 1)

- Rejected – Rezerwacja została odrzucona (istnieje request\_approval o is\_approved = 0)
- Pending – Rezerwacja jeszcze nie została rozpatrzona (nie istnieje request\_approval dla danego rezerwacji)

```
CREATE OR REPLACE FUNCTION REQUEST_STATUS(P_REQUEST_ID NUMBER)
RETURN VARCHAR2
AS
    V_EXISTS NUMBER;
    V_IS_APPROVED NUMBER;
BEGIN
    SELECT COUNT(*) INTO V_EXISTS
    FROM REQUEST_APPROVALS
    WHERE REQUEST_ID = P_REQUEST_ID;
    IF V_EXISTS > 0 THEN
        SELECT IS_APPROVED INTO V_IS_APPROVED
        FROM REQUEST_APPROVALS
        WHERE REQUEST_ID = P_REQUEST_ID;

        IF V_IS_APPROVED = 1 THEN
            RETURN 'Accepted';
        ELSE
            RETURN 'Rejected';
        END IF;
    ELSE
        RETURN 'Pending';
    END IF;
END;
```

#### Przykład:

- request\_status(1) – 'Pending' - jeśli istnieje rezerwacja o id=1, nie istnieje zatwierdzenie danej rezerwacji
- request\_status(1) – 'Accepted' - jeśli istnieje rezerwacja o id=1, istnieje zatwierdzenie danej rezerwacji o is\_approved = 1 (pozytywnie)
- request\_status(1) – 'Rejected' - jeśli istnieje rezerwacja o id=1, istnieje zatwierdzenie danej rezerwacji o is\_approved = 0 (negatywnie)

3. **calculate\_total\_tax(p\_request\_id number)** - oblicza łączny podatek dla rezerwacji domu na podstawie liczby dni wynajmu oraz ceny za dzień. Uwzględnia różne stawki podatkowe w zależności od długości pobytu. ()

```
CREATE OR REPLACE FUNCTION CALCULATE_TOTAL_TAX(P_REQUEST_ID NUMBER)
RETURN NUMBER
```

```

AS
V_DAYS NUMBER;
V_TOTAL_PRICE DECIMAL(7, 2) := 0;
V_PRICE_PER_DAY DECIMAL(7, 2);
V_COTTAGE_ID NUMBER;
V_TOTAL_TAX DECIMAL(7, 2) := 0;
V_FIRST_WEEK_TAX DECIMAL(7, 2) := 0;
V_SECOND_WEEK_TAX DECIMAL(7, 2) := 0;
BEGIN
    SELECT (CHECKOUT_DATE - CHECKIN_DATE), COTTAGE_ID
    INTO V_DAYS, V_COTTAGE_ID
    FROM REQUESTS
    WHERE REQUEST_id = P_REQUEST_ID;

    SELECT MIN_PRICE_PER_DAY
    INTO V_PRICE_PER_DAY
    FROM COTTAGES
    WHERE COTTAGE_ID = V_COTTAGE_ID;

    V_TOTAL_PRICE := V_DAYS * V_PRICE_PER_DAY;
    V_FIRST_WEEK_TAX := 0.36 * 7 + 0.33 * 7 * V_PRICE_PER_DAY;
    V_SECOND_WEEK_TAX := 0.26 * 7 + 0.23 * 7 * V_PRICE_PER_DAY;
    IF V_DAYS = 1 THEN
        V_TOTAL_TAX := 10;
    ELSIF V_DAYS < 7 THEN
        V_TOTAL_TAX := 0.36 * V_DAYS + 0.33 * V_TOTAL_PRICE;
    ELSIF V_DAYS < 14 THEN
        V_TOTAL_TAX := V_FIRST_WEEK_TAX + 0.26 * (V_DAYS - 7) + 0.23 *
(V_DAYS - 7) * V_PRICE_PER_DAY;
    ELSE
        V_TOTAL_TAX := V_FIRST_WEEK_TAX + V_SECOND_WEEK_TAX + 0.15 *
(V_DAYS - 14) + 0.12 * (V_DAYS - 14) * V_PRICE_PER_DAY;
    END IF;

    RETURN V_TOTAL_TAX;
END;

```

- **1 dzień:** Stała wartość podatku wynosi **10**.
- **Mniej niż 7 dni:** Podatek wynosi **0.36** za każdy dzień + **0.33** ceny za każdy dzień.
- **Od 7 do 13 dni:** Podatek za pierwszy tydzień + **0.26** za każdy kolejny dzień + **0.23** ceny za każdy kolejny dzień.
- **14 dni i więcej:** Podatek za pierwszy i drugi tydzień + **0.15** za każdy dodatkowy dzień + **0.12** ceny za każdy dodatkowy dzień.

**Przykład:** istnieje rezerwacja o request\_id = 1, liczba dni = 9, cena za dzień domu = 100

Całkowita cena:  $9 * 100 = 900$ ,

Podatek za 7 dni: 233.52,

Podatek za 2 dni:  $0.26 * 2 + 0.23 * 2 * 100 = 0.52 + 46 = 46.52$

Łącznie: 280.04

4. **avg\_price\_in\_country\_with\_tax(p\_country\_id long)** - oblicza średnią cenę wynajmu domu w kraju o danym country\_id, z uwzględnieniem podatków.

```
CREATE OR REPLACE FUNCTION AVG_PRICE_IN_COUNTRY_WITH_TAX (P_COUNTRY_ID
LONG)
RETURN NUMBER
AS
  V_AVG_PRICE  DECIMAL(7, 2) := 0;
  V_MIN_PRICE  DECIMAL(7, 2) := 0;
  V_TOTAL_PRICE DECIMAL(7, 2) := 0;
BEGIN
  SELECT AVG(TOTAL_PRICE) INTO V_AVG_PRICE
  FROM REQUESTS JOIN COTTAGES USING (COTTAGE_ID)
  JOIN ADDRESSES USING (ADDRESS_ID)
  JOIN CITIES USING (CITY_ID)
  JOIN COUNTRIES USING (COUNTRY_ID)
  WHERE COUNTRY_ID = P_COUNTRY_ID;

  SELECT MIN(MIN_PRICE_PER_DAY) INTO V_MIN_PRICE
  FROM COTTAGES JOIN ADDRESSES USING (ADDRESS_ID)
  JOIN CITIES USING (CITY_ID)
  JOIN COUNTRIES USING (COUNTRY_ID)
  WHERE COUNTRY_ID = P_COUNTRY_ID;

  IF V_MIN_PRICE > 30 THEN
    V_TOTAL_PRICE := V_AVG_PRICE + V_MIN_PRICE;
  ELSIF V_MIN_PRICE > 10 THEN
    V_TOTAL_PRICE := V_AVG_PRICE + V_MIN_PRICE * 2;
  ELSE
    V_TOTAL_PRICE := V_AVG_PRICE;
  END IF;
  RETURN V_TOTAL_PRICE;
END;
```

Pobiera średnią cenę wynajmu (V\_AVG\_PRICE) dla rezerwacji w podanym kraju na podstawie tabel: REQUESTS, COTTAGES, ADDRESSES, CITIES i COUNTRIES.

Pobiera minimalną cenę za dzień (V\_MIN\_PRICE) dla domków w tym kraju.

Na podstawie wartości minimalnej ceny stosuje różne reguły:

- **Jeśli minimalna cena > 30:** Dodaje średnią cenę do minimalnej ceny.
- **Jeśli minimalna cena > 10:** Dodaje średnią cenę do podwójnej minimalnej ceny.

- **W przeciwnym razie:** Zwraca tylko średnią cenę.

Zwraca obliczoną wartość jako średnią cenę wynajmu z uwzględnieniem podatków.

**Przykład:** istnieje kraj o country\_id = 1, średnia cena wynajmu domów = 150, minimalna cena za dzień = 20

$20 > 10$ , więc:  $150 + 20 * 2 = 190$

## Procedury

1. **change\_review(p\_review\_id long, p\_cottage\_id long)** – Zmienia, którego domu dotyczy opinia o danym review\_id.

```
CREATE OR REPLACE PROCEDURE CHANGE_REVIEW (P_REVIEW_ID LONG, P_COTTAGE_ID LONG)
AS
    V_REVIEW NUMBER (4);
BEGIN
    UPDATE REVIEWS
    SET COTTAGE_ID = P_COTTAGE_ID
    WHERE REVIEW_ID = P_REVIEW_ID;
END;
```

2. **exchange\_cottages(first\_cottage\_id long, second\_cottage\_id long)** - zamienia właścicieli dwóch domków oraz ustawia minimalną cenę za dzień dla obu domków na stałą wartość 39.99.

```
CREATE OR REPLACE PROCEDURE EXCHANGE_COTTAGES (FIRST_COTTAGE_ID LONG, SECOND_COTTAGE_ID LONG)
AS
    V_BASE_PRICE DECIMAL(7, 2) := 39.99;
    V_FIRST_OWNER LONG;
    V_SECOND_OWNER LONG;
BEGIN
    SELECT OWNER_ID INTO V_FIRST_OWNER FROM COTTAGES
    WHERE COTTAGE_ID = FIRST_COTTAGE_ID;

    SELECT OWNER_ID INTO V_SECOND_OWNER FROM COTTAGES
    WHERE COTTAGE_ID = SECOND_COTTAGE_ID;

    UPDATE COTTAGES
    SET MIN_PRICE_PER_DAY = V_BASE_PRICE, OWNER_ID = V_SECOND_OWNER
    WHERE COTTAGE_ID = FIRST_COTTAGE_ID;

    UPDATE COTTAGES
    SET MIN_PRICE_PER_DAY = V_BASE_PRICE, OWNER_ID = V_FIRST_OWNER
    WHERE COTTAGE_ID = SECOND_COTTAGE_ID;
END;
```



3. **show\_avg\_price\_per\_user** – Przedstawia zestawienie użytkowników ze średnią ceną za dzień domów do nich należących (używa kursora).

```
CREATE OR REPLACE PROCEDURE SHOW_AVG_PRICE_PER_USER
IS
    CURSOR USER_CURSOR IS
        SELECT AVG(MIN_PRICE_PER_DAY) AS AVG_PRICE, OWNER_ID
        FROM COTTAGES
        GROUP BY OWNER_ID;

    USER_RECORD USER_CURSOR%ROWTYPE;

BEGIN
    -- OPEN USER_CURSOR;

    FOR USER_RECORD IN USER_CURSOR LOOP
        DBMS_OUTPUT.PUT_LINE('OWNER ID: ' || USER_RECORD.OWNER_ID ||
            ' AVG PRICE: ' || USER_RECORD.AVG_PRICE);
    END LOOP;
    -- CLOSE USER_CURSOR;
END SHOW_AVG_PRICE_PER_USER;
```

4. **show\_avg\_grade** – Przedstawia zestawienia domów ze średnią oceną im wystawioną (używa kursora).

```
CREATE OR REPLACE PROCEDURE SHOW_AVG_GRADE
IS
    CURSOR AVERAGE_GRADE_CR IS
        SELECT c.NAME, AVG(r.GRADE) AS AVG_GRADE
        FROM COTTAGES c
        JOIN REVIEWS r ON c.COTTAGE_ID = r.COTTAGE_ID
        GROUP BY c.NAME
        ORDER BY AVG_GRADE DESC;

    V_NAME VARCHAR2(20);
    V_AVG_GRADE NUMBER;

BEGIN
    OPEN AVERAGE_GRADE_CR;
    LOOP
        FETCH AVERAGE_GRADE_CR INTO V_NAME, V_AVG_GRADE;
        EXIT WHEN AVERAGE_GRADE_CR%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE('Cottage: ' || V_NAME || ' - Average Grade: '
            || V_AVG_GRADE);
    END LOOP;

    CLOSE AVERAGE_GRADE_CR;
```

```
END SHOW_AVG_GRADE;
```

## Wyzwalacze

1. **tg\_review\_text** – uzupełnia treść recenzji przed jej wstawieniem do tabeli REVIEWS, jeśli użytkownik nie podał własnego tekstu. Bazuje na ocenie wystawionej dla domu.

```
CREATE OR REPLACE TRIGGER TG_REVIEW_TEXT
BEFORE INSERT ON REVIEWS FOR EACH ROW
when (new.TEXT IS NULL)
DECLARE
    V_COTTAGE_NAME VARCHAR2(50);
BEGIN
    SELECT NAME INTO V_COTTAGE_NAME FROM COTTAGES
        WHERE COTTAGE_ID = :NEW.COTTAGE_ID;

    :NEW.TEXT :=
        CASE :NEW.GRADE
            WHEN 1
            THEN 'Very Bad Cottage.'
            WHEN 2
            THEN 'Bad Cottage.'
            WHEN 3
            THEN 'Decent Cottage.'
            WHEN 4
            THEN 'Good Cottage, I recommend.'
            WHEN 5
            THEN 'Great Cottage, I highly recommend.'
        END;

    dbms_output.put_line('new REVIEW added with grade ' || :NEW.GRADE || '
to cottage ' || V_COTTAGE_NAME);
END;
```

2. **tg\_calculate\_total\_price** – uzupełnia total\_price w rezerwacji, jeśli nie została podana. Oblicza ją na podstawie ilości dni (checkout\_date – checkin\_date) oraz ceny za dzień danego domu.

```
CREATE OR REPLACE TRIGGER TG_CALCULATE_TOTAL_PRICE
BEFORE INSERT OR UPDATE ON REQUESTS
FOR EACH ROW
WHEN (new.TOTAL_PRICE is null)
DECLARE
    V_DAILY_PRICE DECIMAL(7,2);
    V_DAYS NUMBER;
BEGIN
```

```

SELECT MIN_PRICE_PER_DAY INTO V_DAILY_PRICE
FROM COTTAGES
WHERE COTTAGE_ID = :NEW.COTTAGE_ID;

V_DAYS := :NEW.CHECKOUT_DATE - :NEW.CHECKIN_DATE;
:NEW.TOTAL_PRICE := V_DAILY_PRICE * V_DAYS;
END;

```

## Kursory

1. **user\_cursor** – oblicza średnią cenę za dzień dla każdego **owner\_id** z tabeli **cottages**.

```

CURSOR USER_CURSOR IS
    SELECT AVG(MIN_PRICE_PER_DAY) AS AVG_PRICE, OWNER_ID
    FROM COTTAGES
    GROUP BY OWNER_ID;

USER_RECORD USER_CURSOR%ROWTYPE;

```

2. **average\_grade\_cr** – oblicza średnią cenę dla każdego domu.

```

CURSOR AVERAGE_GRADE_CR IS
    SELECT c.NAME, AVG(r.GRADE) AS AVG_GRADE
    FROM COTTAGES c
    JOIN REVIEWS r ON c.COTTAGE_ID = r.COTTAGE_ID
    GROUP BY c.NAME
    ORDER BY AVG_GRADE DESC;

V_NAME VARCHAR2(20);
V_AVG_GRADE NUMBER;

```