



**WYDZIAŁ
ELEKTROTECHNIKI
I INFORMATYKI
POLITECHNIKI RZESZOWSKIEJ**

Interfejs obsługi systemu bazodanowego dla biura podróży - sprawozdanie

Jakub Soboń | Strzępek Katarzyna | Stachiewicz Dawid

173213 | 173220 | 173218

Rzeszów 2025

Spis treści

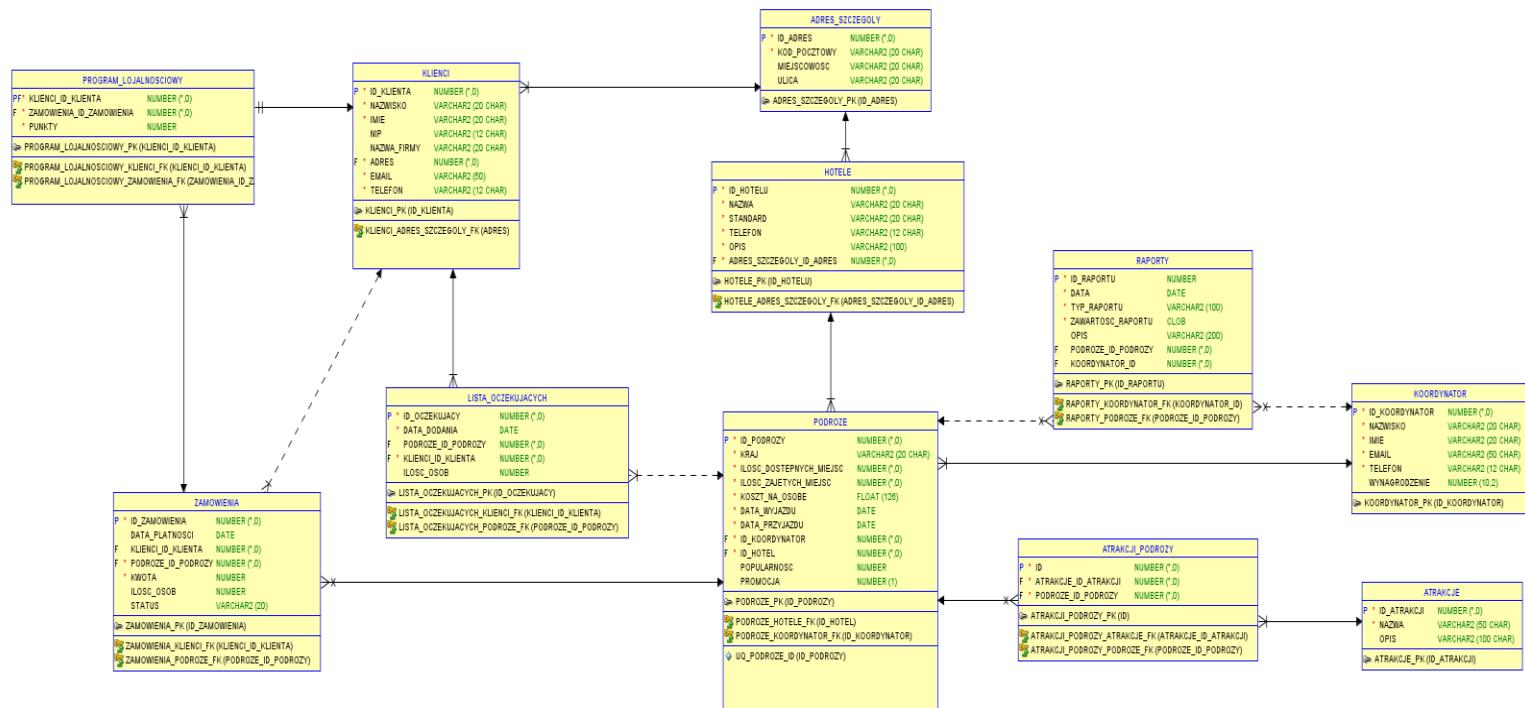
1. Wstęp	4
1.1. Diagram ERD	4
1.2. Logika Aplikacji oraz Diagram Przepływu	5
2. Przedstawienie teoretycznych planów na pakiety, procedury i funkcje.....	6
2.1. Pakiet Klienta.....	6
2.2. Pakiet: Koordynatorzy	19
2.3. Pakiet 3: Analiza i Statystyka	33
3. Apex	47
3.1. Tworzenie tabel	47
3.2. Uzupełnienie danych w tabelach	53
4. Wygląd Aplikacji	55
4.1. Strona Logowania	55
4.2. Strona Główna	56
4.2.1. Struktura strony głównej	57
4.3. Zarządzanie Koordynatorami	58
4.3.1. Struktura strony Koordynatorzy	59
4.4. Strona Zarządzanie klientami	63
4.4.1. Struktura strony Zarządzanie Klientami.....	65
4.5. Strona Zarządzaj wycieczkami	66
4.5.1. Struktura strony zarządzaj wycieczkami	68
4.6. Strona Zarządzaj Hotelami	70
4.6.1. Struktura strony Zarządzaj hotelami	71
4.7. Strona Statystyki	73
4.7.1. Struktura strony	74
4.8. Strona Zarządzaj Rezerwacjami	76
4.8.1. Struktura zarządzaj rezerwacjami.....	77
4.9. Strona Płatności	79
4.9.1. Struktura strony płatności.....	80
4.10. Strona Program Lojalnościowy	82
4.10.1. Struktura strony Program Lojalnościowy.....	83
4.11. Strona Podsumowanie Miesiąca	85
4.11.1. Struktura strony Podsumowanie miesiąca.....	86
5. Export i Import	88
5.1. Export 88	
5.2. Import 90	

1. Wstęp

System bazodanowy został zaprojektowany jako modelowy przykład działania systemu rezerwacyjnego dla biura podróży. Jego struktura opiera się na 11 tabelach oraz trzech pakietach wspierających realizację kluczowych procesów biznesowych. W bazie danych przechowywane są informacje o klientach, wycieczkach, hotelach i rezerwacjach, co umożliwia kompleksowe zarządzanie operacjami związanymi z organizacją podróży.

Projekt wyróżnia się zaawansowanymi funkcjonalnościami, takimi jak efektywne zarządzanie rezerwacjami, dynamiczne przypisywanie hoteli do wycieczek oraz generowanie szczegółowych raportów dotyczących sprzedaży i efektywności pracy koordynatorów. Dzięki temu system zapewnia elastyczne narzędzie wspierające zarówno codzienną pracę biura podróży, jak i analizę wyników działalności.

1.1. Diagram ERD

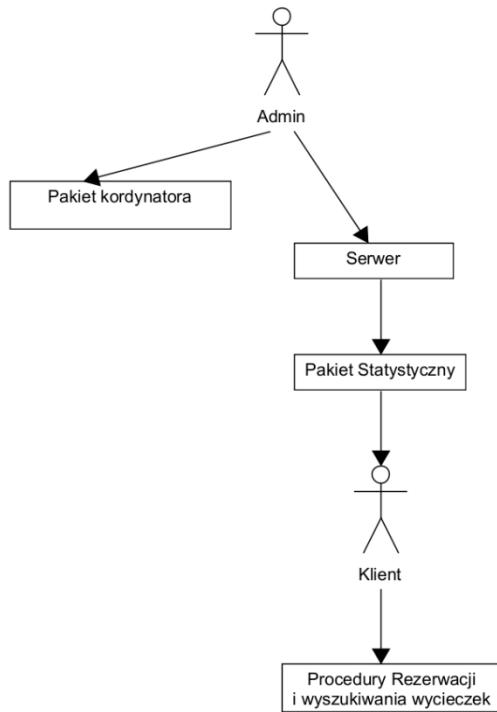


Rysunek 1 Diagram ERD Aplikacji

Diagram ERD przedstawia strukturę bazy danych systemu rezerwacyjnego dla biura podróży. Kluczowe tabele, takie jak KLIENCI, PODROZE, HOTELE i REZERWACJE, przechowują informacje o klientach, organizowanych wycieczkach, dostępnych hotelach i dokonanych rezerwacjach. Relacje między tabelami zapewniają integralność danych – na przykład tabela REZERWACJE łączy klientów z konkretnymi wycieczkami oraz dodatkowymi usługami. Dzięki zastosowaniu kluczy głównych i obcych system umożliwia efektywne zarządzanie danymi oraz ich szybkie przetwarzanie.

1.2. Logika Aplikacji oraz Diagram Przepływu

Logika aplikacji wygląda następująco:



Rysunek 2 Logika Aplikacji

Diagram przepływu danych przedstawia kluczowe interakcje między użytkownikami systemu – adminem i klientem – a modułami aplikacji, takimi jak pakiet koordynatora i pakiet statystyczny. Struktura ta umożliwia skuteczne zarządzanie operacjami oraz analizę danych w systemie rezerwacyjnym.

Pakiet koordynatora odpowiada za zarządzanie procesami organizacyjnymi, w tym przypisywanie klientów do wycieczek, zarządzanie hotelami oraz dodatkowymi atrakcjami. Umożliwia również aktualizację szczegółowych danych uczestników wycieczek oraz monitorowanie pracy koordynatorów, co wspiera efektywne działanie systemu.

Pakiet statystyczny zapewnia narzędzia do generowania raportów dotyczących sprzedaży, obłożenia hoteli, efektywności pracy koordynatorów oraz przychodów. Raporty te wspierają administratora w podejmowaniu decyzji i optymalizacji procesów.

Klient ma dostęp do procedur związanych z wyszukiwaniem i rezerwacją wycieczek. Może przeglądać dostępne oferty, sprawdzać terminy oraz dostępność miejsc. W przypadku braku miejsc system dynamicznie zarządza listą oczekujących, co zwiększa szanse na realizację przyszłych rezerwacji.

Dzięki integracji pakietów i procedur system skutecznie wspiera zarówno klientów, jak i administratorów, umożliwiając sprawną obsługę kluczowych operacji oraz optymalizację działania całego systemu.

2. Przedstawienie teoretycznych planów na pakiety, procedury i funkcje

Funkcje w projekcie Oracle APEX to elementy logiki biznesowej implementowane w PL/SQL, które przetwarzają dane i zwracają konkretne wartości, takie jak liczby, teksty lub zestawy wyników. Są używane do obliczeń, walidacji danych oraz generowania raportów, wspierając działanie aplikacji.

Procedura to blok kodu w języku programowania bazodanowego, takim jak PL/SQL, który realizuje określone zadanie lub zestaw zadań. W przeciwieństwie do funkcji, procedura nie zwraca bezpośrednio wartości, ale może modyfikować dane w bazie, wykonywać operacje walidacyjne lub wywoływać inne procedury. Procedury są zwykle używane do wykonywania bardziej złożonych procesów biznesowych w bazach danych.

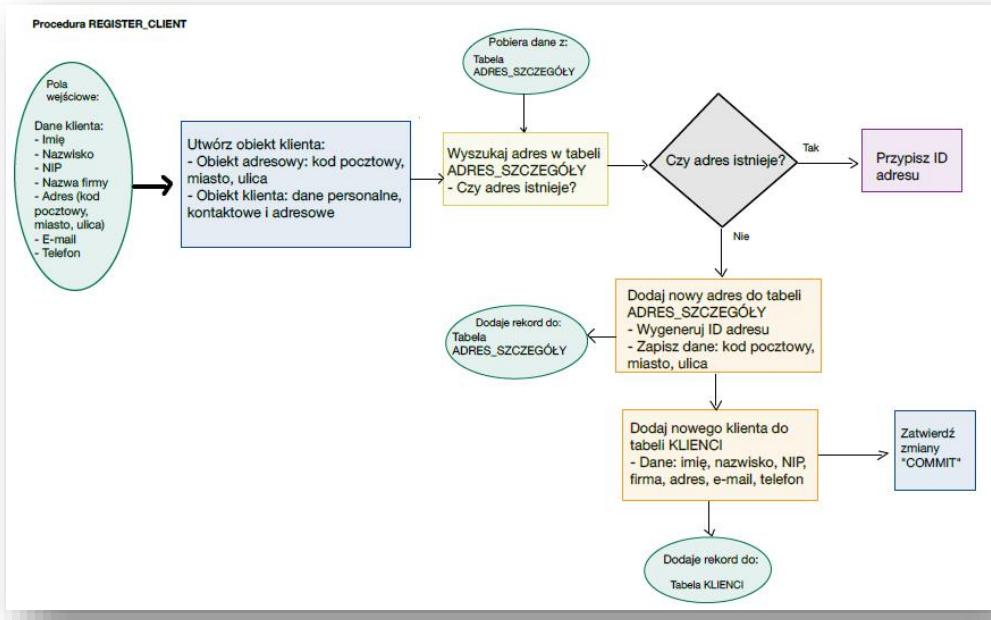
2.1. Pakiet Klienta

Pakiet nr 1 jest podstawowym elementem systemu zarządzania biurem podróży. Jego głównym celem jest obsługa kluczowych operacji związanych z klientami, podróżami i rezerwacjami. Pakiet ten zawiera zestaw procedur i funkcji umożliwiających realizację takich zadań jak rejestracja klientów, wyszukiwanie wycieczek, zarządzanie zamówieniami, realizacja płatności oraz przetwarzanie danych dotyczących podróży.

Pakiet jest zaprojektowany w oparciu o logikę biznesową, dzięki czemu umożliwia efektywną i bezpieczną obsługę procesów. Wykorzystuje mechanizmy obsługi błędów, kursory, sekwencje i transakcje, zapewniając integralność danych.

REGISTER_CLIENT

Procedura służy do rejestracji nowego klienta w systemie. Sprawdza, czy adres klienta już istnieje w bazie danych, a jeśli nie, dodaje nowy adres do tabeli ADRES_SZCZEGOLY. Następnie zapisuje dane klienta w tabeli KLIENCI.



```

create or replace PROCEDURE REGISTER_CLIENT (
    p_name IN VARCHAR2,
    p_first_name IN VARCHAR2,
    p_nip IN VARCHAR2,
    p_company_name IN VARCHAR2,
    p_address IN VARCHAR2,
    p_postal_code IN VARCHAR2,
    p_city IN VARCHAR2,
    p_email IN VARCHAR2,
    p_phone IN VARCHAR2
) IS
    -- Obiekty
    v_klient typ_klient;
    v_address typ_adres := typ_adres(p_postal_code, p_city, p_address);

    -- Kursor
    CURSOR c_address IS
        SELECT ID_ADRES
        FROM ADRES_SZCZEGOLY
        WHERE KOD_POCZTOWY = p_postal_code
            AND MIEJSCOWOSC = p_city
            AND ULICA = p_address;

    v_address_id NUMBER;
BEGIN
    -- Utworzenie obiektu klienta
    v_klient := typ_klient(p_name, p_first_name, p_nip, p_company_name, v_address, p_email, p_phone);

    -- Wyszukiwanie adresu
    OPEN c_address;
    FETCH c_address INTO v_address_id;

    IF c_address%NOTFOUND THEN
        -- Jeśli adres nie istnieje, dodajemy go
        INSERT INTO ADRES_SZCZEGOLY (ID_ADRES, KOD_POCZTOWY, MIEJSCOWOSC, ULICA)
        VALUES (ADRES_SEQ.NEXTVAL, p_postal_code, p_city, p_address);
        v_address_id := ADRES_SEQ.CURRVAL;
    END IF;

    CLOSE c_address;

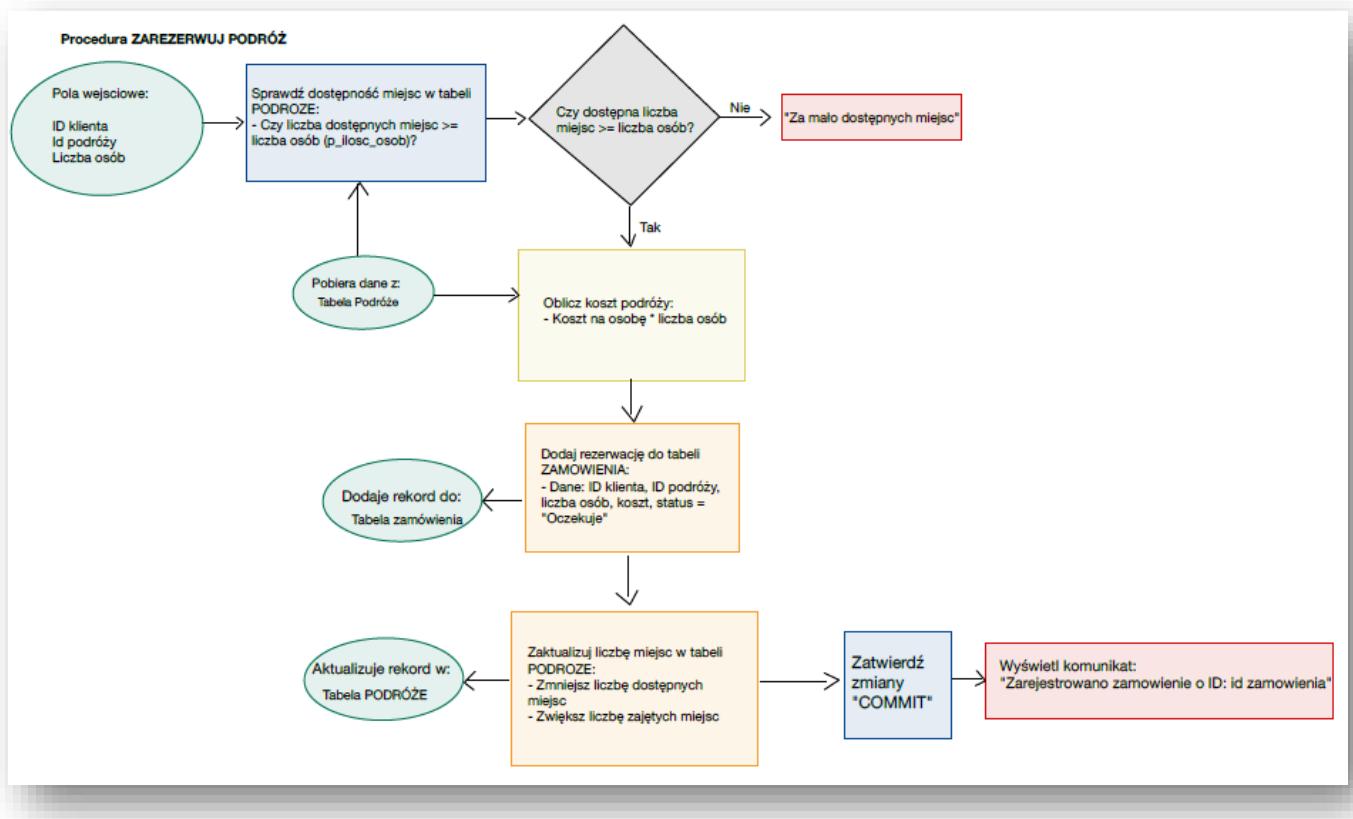
    -- Dodawanie klienta
    INSERT INTO KLIENCI (ID_KLIENCI, NAZWISKO, IMIE, NIP, NAZWA_FIRMY, ADRES, EMAIL, TELEFON)
    VALUES (KLIENCI_SEQ.NEXTVAL, v_klient.nazwisko, v_klient.imie, v_klient.nip, v_klient.firma, v_address_id, v_klient.email, v_klient.telefon);

    COMMIT;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20001, 'Nie znaleziono adresu.');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20002, 'Nieoczekiwany błąd: ' || SQLERRM);
END REGISTER_CLIENT;

```

ZAREZERWUJ PODROZ

Procedura umożliwia rezerwację wycieczki przez klienta. Przeprowadza sprawdzenie dostępności miejsc, oblicza koszt rezerwacji, dodaje wpis do tabeli zamówień, a następnie aktualizuje liczbę dostępnych i zajętych miejsc w wybranej podróży.



```

create or replace PROCEDURE ZAREZERWUJ PODROZ (
    p_id_klienta IN NUMBER,          -- ID klienta
    p_id_podrozy IN NUMBER,          -- ID podrózy
    p_ilosc_osob IN NUMBER          -- Liczba osób
) IS
    v_koszt_podrozy NUMBER;          -- Koszt podrózy
    v_id_zamowienia NUMBER;          -- ID zamówienia
BEGIN
    -- 1. Sprawdzenie dostępności miejsc w podróży
    SELECT ILOSC_DOSTEPNYCH_MIEJSC
    INTO v_koszt_podrozy
    FROM PODROZE
    WHERE ID_PODROZY = p_id_podrozy;

    IF v_koszt_podrozy < p_ilosc_osob THEN
        RAISE_APPLICATION_ERROR(-20001, 'Za mało dostępnych miejsc w wycieczce.');
    END IF;

    -- 2. Obliczenie kosztu podrózy
    SELECT KOSZT_NA_OSOBE * p_ilosc_osob
    INTO v_koszt_podrozy
    FROM PODROZE
    WHERE ID_PODROZY = p_id_podrozy;

    -- 3. Dodanie rezerwacji do tabeli `ZAMOWIENIA`
    INSERT INTO ZAMOWIENIA (
        ID_ZAMOWIENIA,
        KLIENCI_ID_KLIENTA,
        PODROZE_ID_PODROZY,
        ILOSC_OSOB,
        KWOTA,
        STATUS
    )
    VALUES (
        ZAMOWIENIA_SEQ.NEXTVAL,
        p_id_klienta,
        p_id_podrozy,
        p_ilosc_osob,
        v_koszt_podrozy,
        'Oczekuje'
    )
    RETURNING ID_ZAMOWIENIA INTO v_id_zamowienia;

    -- 4. Aktualizacja liczby miejsc w podróży
    UPDATE PODROZE
    SET ILOSC_DOSTEPNYCH_MIEJSC = ILOSC_DOSTEPNYCH_MIEJSC - p_ilosc_osob,
        ILOSC_ZAJETYCH_MIEJSC = ILOSC_ZAJETYCH_MIEJSC + p_ilosc_osob
    WHERE ID_PODROZY = p_id_podrozy;

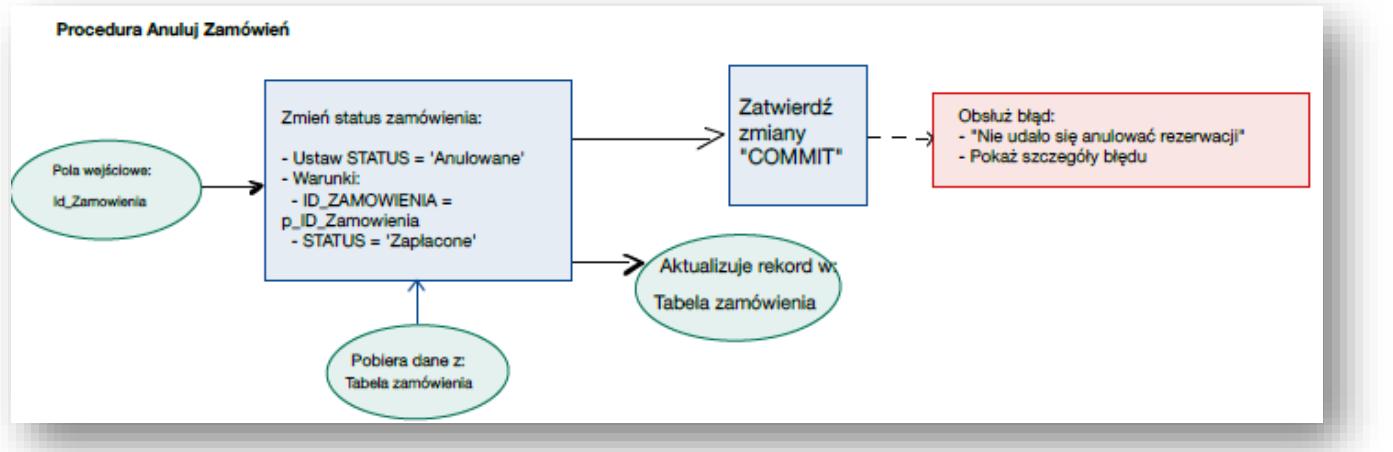
    -- Zatwierdzenie transakcji
    COMMIT;

    -- Informacja o sukcesie
    DBMS_OUTPUT.PUT_LINE('Zarejestrowano zamówienie o ID: ' || v_id_zamowienia);
EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
        RAISE_APPLICATION_ERROR(-20099, 'Wystąpił nieoczekiwany błąd: ' || SQLERRM);
END;
/

```

ANULUJ_ZAMOWIENIE

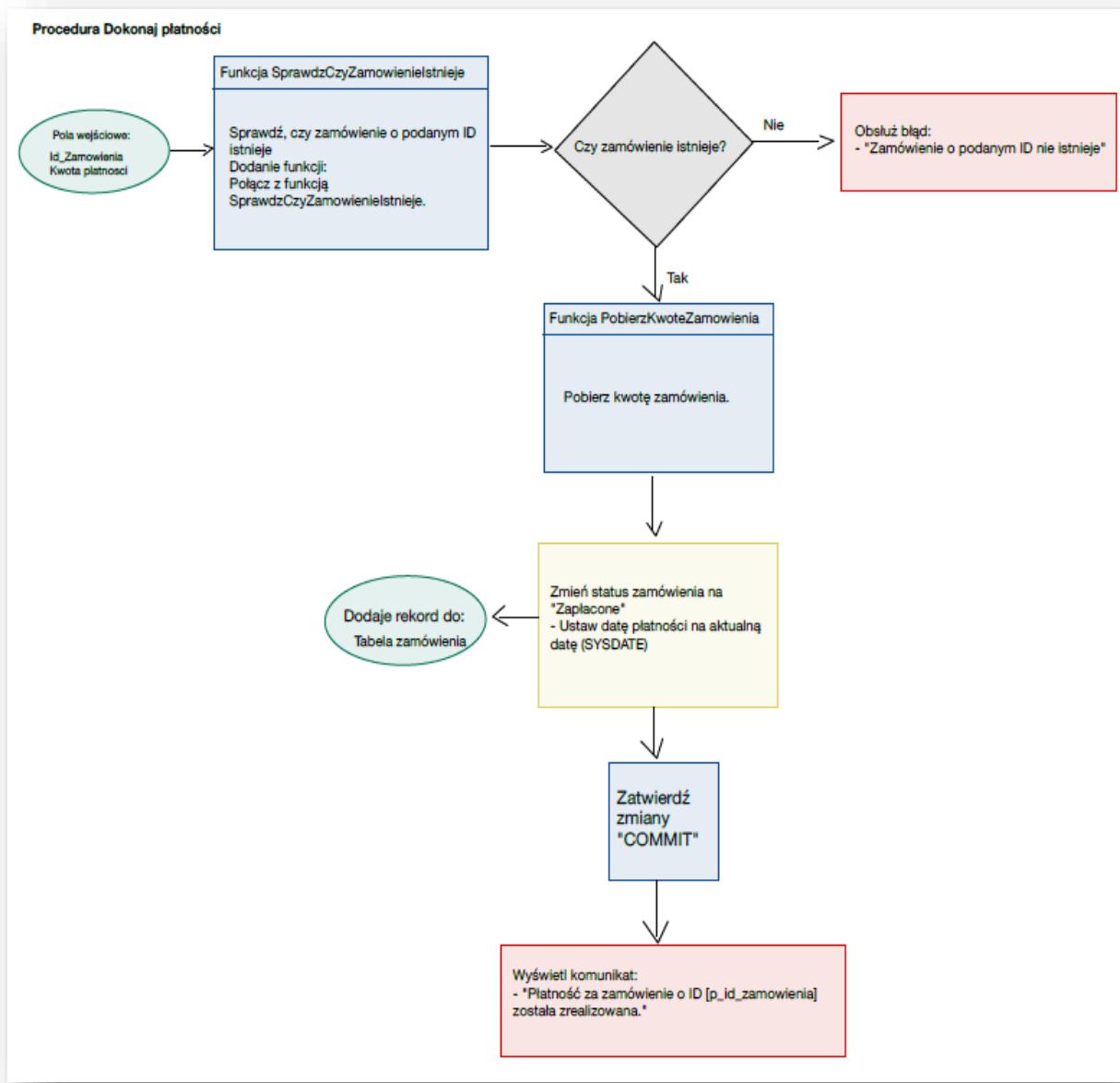
Procedura służy do anulowania zamówienia w systemie. Przyjmuje jako parametr identyfikator zamówienia (p_ID_Zamowienia). Sprawdza, czy status zamówienia jest ustawiony na "Zapłacone". Jeśli warunek jest spełniony, zmienia status zamówienia na "Anulowane" i zatwierdza zmiany w bazie danych. W przypadku wystąpienia błędu procedura zgłasza wyjątek z informacją o niepowodzeniu anulowania rezerwacji.



```
1  create or replace PROCEDURE AnulujZamowienie (
2      p_ID_Zamowienia IN NUMBER
3  ) AS
4  BEGIN
5      -- Zmień status zamówienia na "Anulowane"
6      UPDATE ZAMOWIENIA
7          SET STATUS = 'Anulowane'
8      WHERE ID_ZAMOWIENIA = p_ID_Zamowienia
9          AND STATUS = 'Zapłacone';
10
11     -- Zatwierdź zmiany
12     COMMIT;
13 EXCEPTION
14     WHEN OTHERS THEN
15         RAISE_APPLICATION_ERROR(-20001, 'Nie udało się anulować rezerwacji: ' || SQLERRM);
16 END;
17 /
```

DOKONAJ_PLATNOSCI

Procedura umożliwia realizację płatności za zamówienie. Przyjmuje dwa parametry: identyfikator zamówienia (p_id_zamowienia) oraz kwotę płatności (p_kwota). Wykonuje następujące operacje:



```
1  create or replace PROCEDURE DOKONAJ_PLATNOSCI (
2    p_id_zamowienia IN NUMBER,
3    p_kwota IN NUMBER
4  ) IS
5    v_kwota_zamowienia NUMBER;
6  BEGIN
7    -- 1. Sprawdzenie, czy zamówienie istnieje
8    IF NOT SprawdzCzyZamowienieIstnieje(p_id_zamowienia) THEN
9      RAISE_APPLICATION_ERROR(-20002, 'Zamówienie o podanym ID nie istnieje.');
10   END IF;
11
12   -- 2. Pobranie kwoty zamówienia
13   v_kwota_zamowienia := PobierzKwoteZamowienia(p_id_zamowienia);
14
15
16   -- 4. Aktualizacja statusu zamówienia
17   UPDATE ZAMOWIENIA
18     SET STATUS = 'Zapłacone',
19         DATA_PLATNOSCI = SYSDATE
20   WHERE ID_ZAMOWIENIA = p_id_zamowienia;
21
22   COMMIT;
23
24   DBMS_OUTPUT.PUT_LINE('Płatność za zamówienie o ID ' || p_id_zamowienia || ' została zrealizowana.');
25
26 EXCEPTION
27   WHEN OTHERS THEN
28     ROLLBACK;
29     RAISE_APPLICATION_ERROR(-20099, 'Błąd podczas obsługi płatności: ' || SQLERRM);
30 END DOKONAJ_PLATNOSCI;
31 /
```

Trigger TRG_AKTUALIZUJ_DATE_PLATNOSCI

```
1  create or replace TRIGGER "TRG_AKTUALIZUJ_DATE_PLATNOSCI"
2  BEFORE UPDATE OF STATUS ON ZAMOWIENIA
3  FOR EACH ROW
4  WHEN (NEW.STATUS = 'Zapłacone')
5  BEGIN
6    -- Ustawienie daty płatności na bieżącą datę
7    :NEW.DATA_PLATNOSCI := SYSDATE;
8  END;
9  /
```

Trigger TRG_AKTUALIZUJ_DATE_PLATNOSCI



Jak działa logika triggera w praktyce?

1. Aktualizacja zamówienia

Pracownik lub system dokonuje aktualizacji statusu zamówienia w tabeli ZAMOWIENIA.
Przykład: Zmiana statusu na „Zapłacone”.

2. Uruchomienie triggera

Gdy status zamówienia zostaje zaktualizowany, trigger się aktywuje.

Warunek działania: Trigger sprawdza, czy nowy status zamówienia to „Zapłacone”. Jeśli tak, przechodzi do kolejnych kroków.

3. Ustawienie daty płatności

Jeśli warunek jest spełniony, trigger automatycznie ustawia wartość kolumny DATA_PLATNOSCI na bieżącą datę (SYSDATE).

Efekt: W tabeli ZAMOWIENIA dla tego zamówienia zapisuje się aktualna data jako data płatności.

4. Koniec działania

Jeśli warunek (STATUS = „Zapłacone”) nie jest spełniony, trigger kończy swoje działanie bez wprowadzania zmian w tabeli.

Przykład: Jeśli status zmieni się na „Anulowane” lub inny, trigger nie podejmie żadnych działań.

Przykłady praktyczne

Sytuacja 1: Zmiana statusu na „Zapłacone”

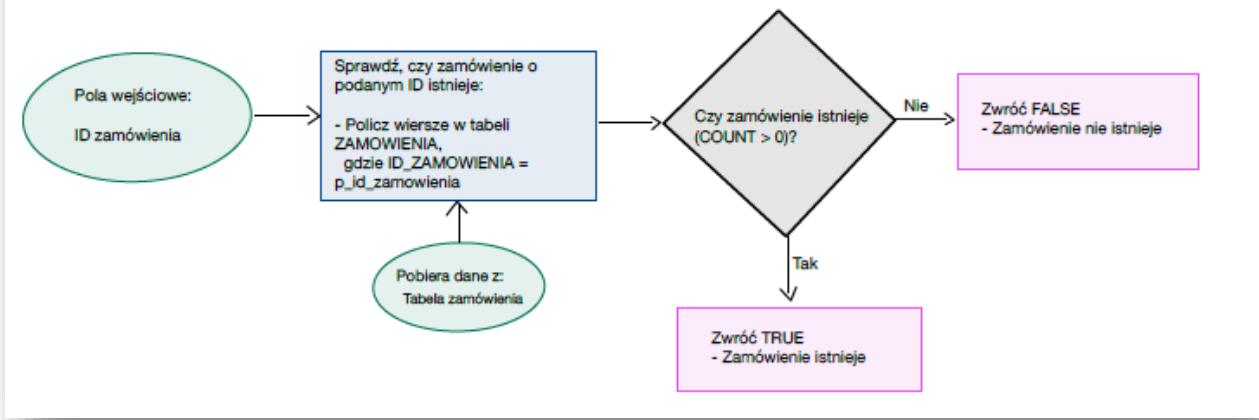
Pracownik zmienia status zamówienia na „Zapłacone”.
Trigger się aktywuje, sprawdza, czy status spełnia warunek (STATUS = „Zapłacone”).
Kolumna DATA_PLATNOSCI w tabeli ZAMOWIENIA zostaje ustawiona na aktualną datę.
Efekt końcowy: Zamówienie ma zaktualizowany status i zapisano datę płatności.

Sytuacja 2: Zmiana statusu na „Anulowane”

Pracownik zmienia status zamówienia na „Anulowane”.
Trigger sprawdza warunek, ale ponieważ nowy status to nie „Zapłacone”, trigger kończy działanie bez wprowadzania zmian.
Efekt końcowy: Trigger nie modyfikuje tabeli ZAMOWIENIA.

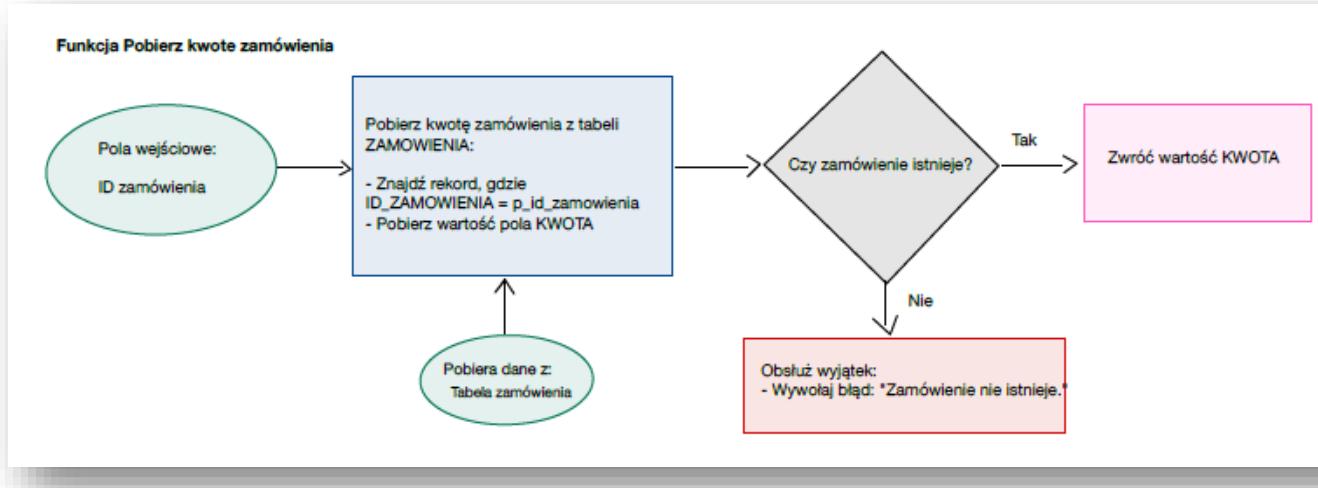
FUNKCJA Sprawdź czy zamówienie istnieje

Funkcja Sprawdź czy zamówienie istnieje



```
1  create or replace FUNCTION SprawdzCzyZamowienieIstnieje (
2      p_id_zamowania IN NUMBER
3  ) RETURN BOOLEAN IS
4      v_exists NUMBER;
5  BEGIN
6      SELECT COUNT(*)
7      INTO v_exists
8      FROM ZAMOWIENIA
9      WHERE ID_ZAMOWIENIA = p_id_zamowania;
10     RETURN v_exists > 0;
11 END SprawdzCzyZamowienieIstnieje;
12 /
```

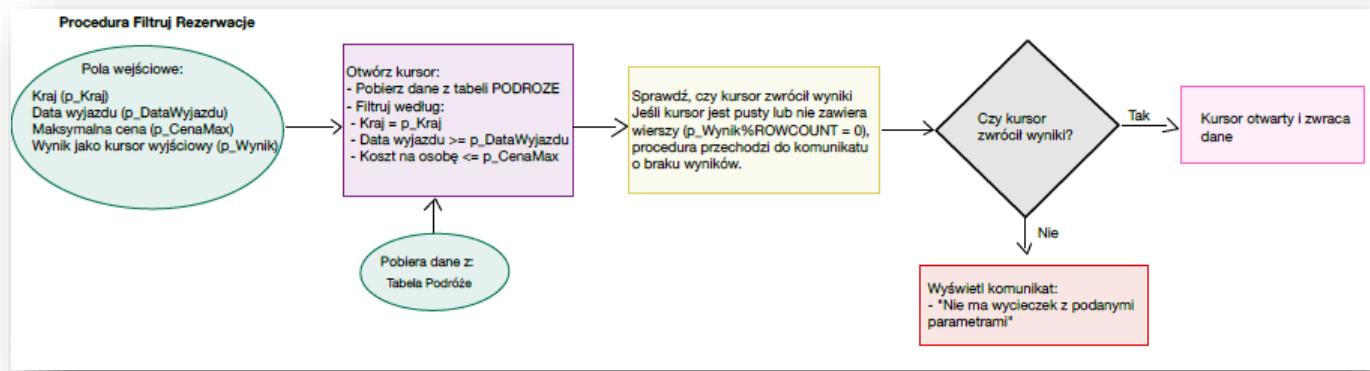
FUNKCJA Pobierz kwotę zamówienia



```
1 create or replace FUNCTION PobierzKwoteZamowienia (
2   |   p_id_zamowienia IN NUMBER
3   ) RETURN NUMBER IS
4   |   v_kwota NUMBER;
5 BEGIN
6   |   SELECT KWOTA
7   |   INTO v_kwota
8   |   FROM ZAMOWIENIA
9   |   WHERE ID_ZAMOWIENIA = p_id_zamowienia;
10  |   RETURN v_kwota;
11 EXCEPTION
12   |   WHEN NO_DATA_FOUND THEN
13   |       RAISE_APPLICATION_ERROR(-20002, 'Zamówienie nie istnieje.');
14 END PobierzKwoteZamowienia;
15 /
```

FILTRUJ_REZERWACJE

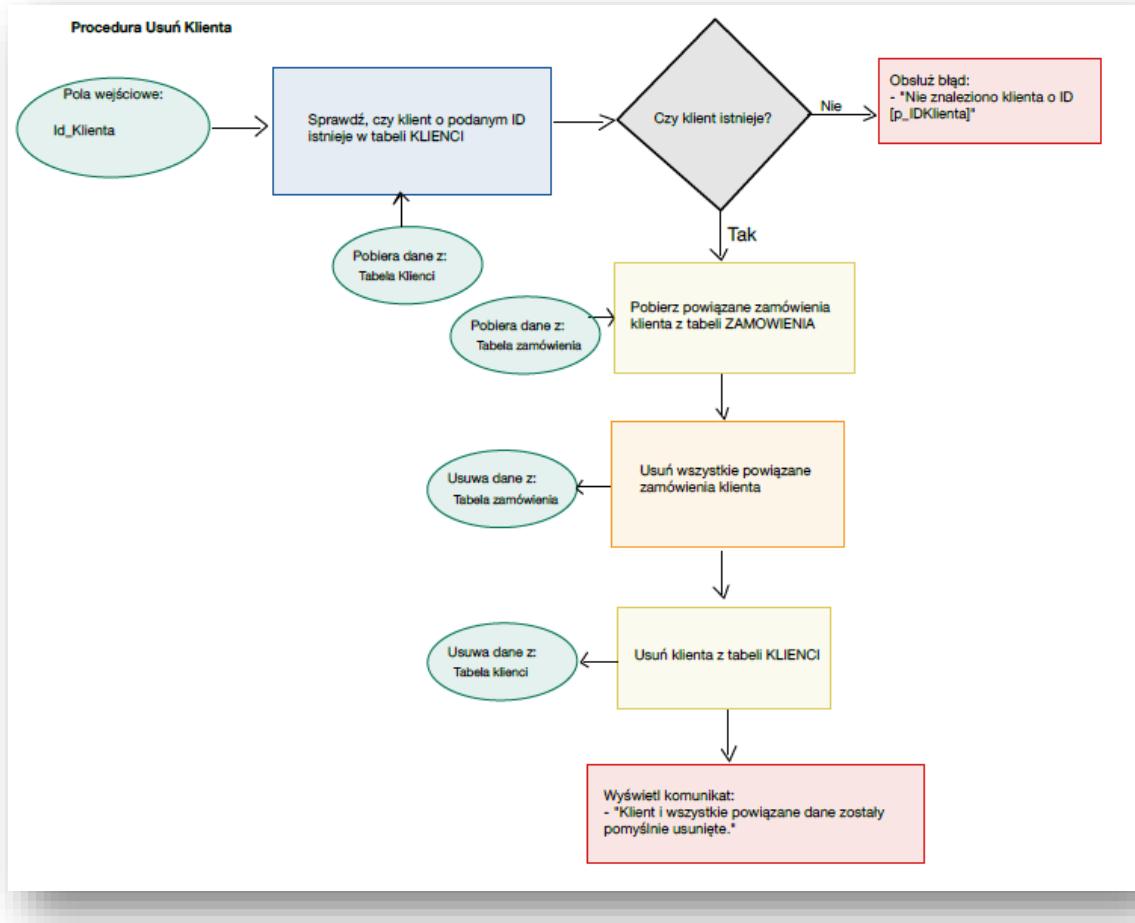
Procedura służy do filtrowania dostępnych wycieczek na podstawie podanych kryteriów. Przyjmuje jako parametry: kraj wyjazdu (p_Kraj), minimalną datę wyjazdu (p_DataWyjazdu), maksymalną cenę za osobę (p_CenaMax) oraz zmienną wyjściową (p_Wynik) będącą kursem referencyjnym (SYS_REF_CURSOR).



```
1  create or replace PROCEDURE FiltrujRezerwacje(
2      p_Kraj IN VARCHAR2,
3      p_DataWyjazdu IN DATE,
4      p_CenaMax IN FLOAT,
5      p_Wynik OUT SYS_REF_CURSOR
6  ) IS
7  BEGIN
8      -- Otwórz kurser z danymi
9      OPEN p_Wynik FOR
10     SELECT *
11        FROM PODROZE
12     WHERE KRAJ = p_Kraj
13        AND DATA_WYJAZDU >= p_DataWyjazdu
14        AND KOSZT_NA_OSOBE <= p_CenaMax;
15
16     -- Jeśli kurser jest pusty, poinformuj użytkownika
17    IF NOT p_Wynik%ISOPEN OR p_Wynik%ROWCOUNT = 0 THEN
18        DBMS_OUTPUT.PUT_LINE('Nie ma wycieczek z podanymi parametrami.');
19    END IF;
20  END FiltrujRezerwacje;
21  /
```

USUN_Klienta

Procedura umożliwia usunięcie klienta z bazy danych wraz z wszystkimi jego powiązanyimi zamówieniami.



```
1  create or replace PROCEDURE UsunKlienta (
2      p_IDKlienta IN NUMBER
3  ) IS
4      -- Kolekcja zamówień
5      TYPE lista_zamówień IS TABLE OF ZAMÓWIENIA%ROWTYPE;
6
7      v_zamówienia lista_zamówień;
8
9      -- Obiekt klienta
10     TYPE typ_klient IS RECORD (
11         id_klienta NUMBER,
12         nazwisko    VARCHAR2(255),
13         imie        VARCHAR2(255)
14     );
15     v_klient typ_klient;
16
17     BEGIN
18         -- Walidacja: sprawdzenie, czy klient istnieje
19         SELECT ID_KLIENTA, NAZWISKO, IMIE
20             INTO v_klient.id_klienta, v_klient.nazwisko, v_klient.imie
21             FROM KLIENCI
22             WHERE ID_KLIENTA = p_IDKlienta;
23
24         -- Pobranie powiązanych zamówień
25         SELECT * BULK COLLECT INTO v_zamówienia
26             FROM ZAMÓWIENIA
27             WHERE KLIENCI_ID_KLIENTA = p_IDKlienta;
28
29         -- Usunięcie zamówień
30         FORALL i IN v_zamówienia.FIRST .. v_zamówienia.LAST
31             DELETE FROM ZAMÓWIENIA WHERE ID_ZAMÓWIENIA = v_zamówienia(i).ID_ZAMÓWIENIA;
32
33         -- Usunięcie klienta
34         DELETE FROM KLIENCI WHERE ID_KLIENTA = p_IDKlienta;
35
36         -- Potwierdzenie operacji
37         DBMS_OUTPUT.PUT_LINE('Klient i wszystkie powiązane dane zostały pomyślnie usunięte.');
38
39     EXCEPTION
40         WHEN NO_DATA_FOUND THEN
41             DBMS_OUTPUT.PUT_LINE('Nie znaleziono klienta o ID ' || p_IDKlienta || '.');
42         WHEN OTHERS THEN
43             DBMS_OUTPUT.PUT_LINE('Wystąpił błąd: ' || SQLERRM);
44             ROLLBACK;
45     END UsunKlienta;
```

2.2. Pakiet: Koordynatorzy

Pakiet Koordynatorzy został zaprojektowany jako kluczowy element systemu zarządzania bazą danych biura podróży. Pakiet ten skupia się na zarządzaniu danymi związanymi z koordynatorami, podróżami, hotelami i atrakcjami turystycznymi. Dzięki odpowiednio zaimplementowanym procedurom i funkcjom, pakiet umożliwia efektywne zarządzanie kluczowymi operacjami, takimi jak dodawanie nowych koordynatorów, organizowanie wycieczek, przypisywanie hoteli oraz anulowanie podróży.

Pakiet wykorzystuje mechanizmy takie jak kursory, funkcje, obsługa wyjątków oraz sekwencje, co pozwala na bezpieczne, automatyczne i wydajne przetwarzanie danych.

Cel pakietu

Celem Pakietu 2: Koordynatorzy jest usprawnienie zarządzania danymi i operacjami związanymi z organizacją wycieczek i pracą koordynatorów w biurze podróży. Główne cele pakietu obejmują:

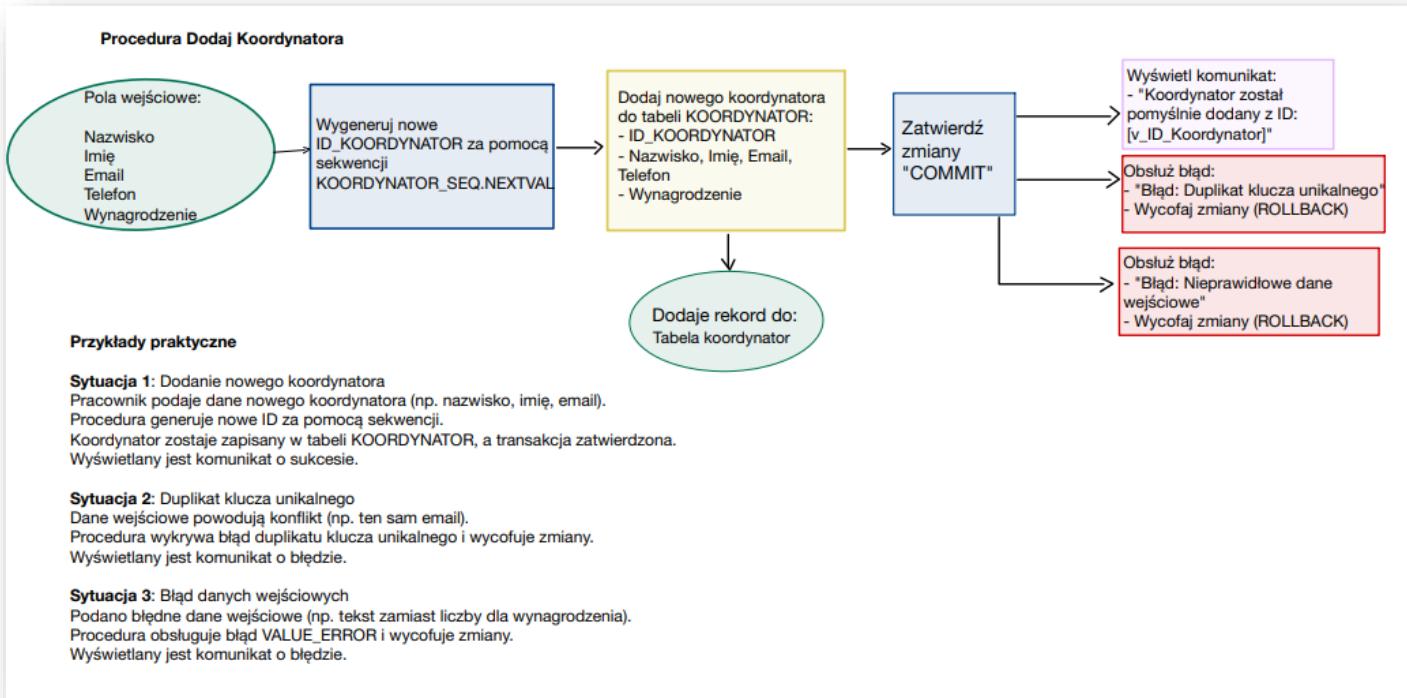
- 1) Automatyzację procesów dodawania i zarządzania danymi koordynatorów, podróży i hoteli.
- 2) Usprawnienie procesu tworzenia nowych wycieczek poprzez integrację z tabelami podróży, atrakcji i hoteli.
- 3) Umożliwienie bezpiecznego anulowania wycieczek oraz powiadamiania klientów.
- 4) Zastosowanie mechanizmów obsługi błędów dla zapewnienia stabilności i niezawodności systemu.

Opis procedur

1. Dodaj Koordynatora

Cel: Dodaje nowego koordynatora do tabeli KOORDYNATOR. Generuje unikalne ID dla koordynatora za pomocą sekwencji KOORDYNATOR_SEQ.

Logika: Procedura przyjmuje dane wejściowe takie jak nazwisko, imię, e-mail, telefon i wynagrodzenie, a następnie wstawia nowy rekord do tabeli KOORDYNATOR. Obsługuje błędy związane z duplikatami i nieprawidłowymi danymi.

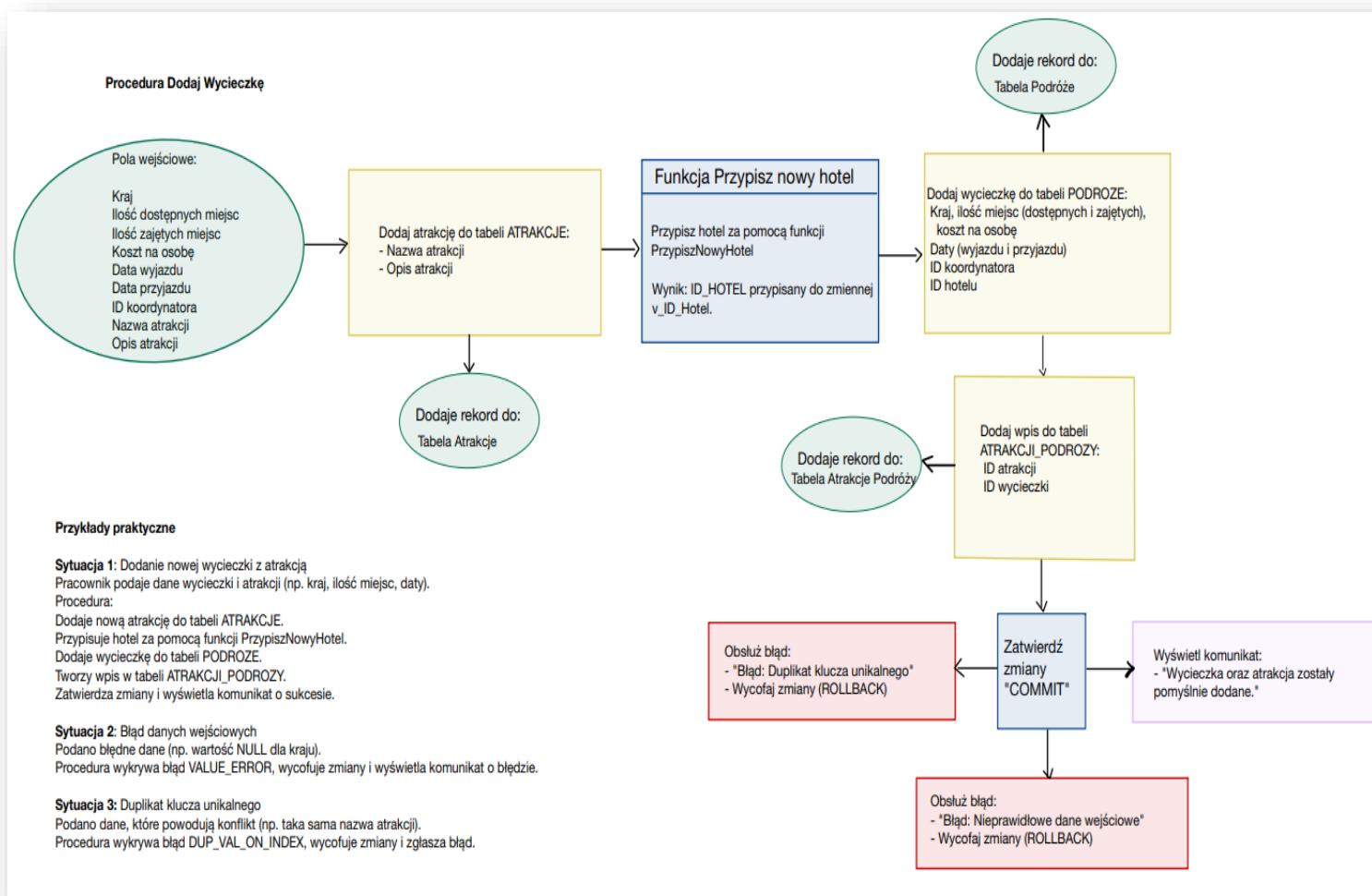


```

1  < create or replace PROCEDURE DodajKoordynator(
2      p_Nazwisko IN VARCHAR2,
3      p_Imie IN VARCHAR2,
4      p_Email IN VARCHAR2,
5      p_Telefon IN VARCHAR2,
6      p_Wynagrodzenie IN NUMBER
7  ) IS
8      v_ID_Koordynator NUMBER; -- Zmienna do przechowania ID Koordynatora
9  BEGIN
10     -- Generowanie ID_KOORDYNATOR za pomocą sekwencji
11     v_ID_Koordynator := KOORDYNATOR_SEQ.NEXTVAL;
12
13     -- Wstawienie nowego rekordu do tabeli KOORDYNATOR
14     INSERT INTO KOORDYNATOR (
15         ID_KOORDYNATOR,
16         NAZWISKO,
17         IMIE,
18         EMAIL,
19         TELEFON,
20         WYNAGRODZENIE
21     ) VALUES (
22         v_ID_Koordynator,
23         p_Nazwisko,
24         p_Imie,
25         p_Email,
26         p_Telefon,
27         p_Wynagrodzenie
28     );
29
30     -- Zatwierdzenie transakcji
31     COMMIT;
32
33     -- Komunikat o sukcesie
34     DBMS_OUTPUT.PUT_LINE('Koordynator został pomyślnie dodany z ID: ' || v_ID_Koordynator);
35
36     EXCEPTION
37     WHEN DUP_VAL_ON_INDEX THEN
38         -- Obsługa błędu duplikatu klucza unikalnego
39         ROLLBACK;
40         RAISE_APPLICATION_ERROR(-20001, 'Błąd: Duplikat klucza unikalnego.');
41
42     WHEN VALUE_ERROR THEN
43         -- Obsługa błędu wartości (np. zbyt duży tekst lub błędne dane wejściowe)
44         ROLLBACK;
45         RAISE_APPLICATION_ERROR(-20002, 'Błąd: Nieprawidłowe dane wejściowe.');
46
47     WHEN OTHERS THEN
48         -- Obsługa pozostałych błędów
49         ROLLBACK;
50         RAISE_APPLICATION_ERROR(-20003, 'Błąd podczas dodawania koordynatora: ' || SQLERRM);
51     END DodajKoordynator;
52 
```

2. DodajWycieczke

Cel: Dodaje nową wycieczkę oraz związaną z nią atrakcję do bazy danych.
Logika: Procedura tworzy nowy rekord w tabeli ATRAKCJE, przypisuje hotel za pomocą funkcji PrzypiszNowyHotel, a następnie dodaje nową podróż do tabeli PODROZE. Rekordy są łączone w tabeli pośredniej ATRAKCJI PODROZY. Wykorzystuje sekwencje ATRAKCJE_SEQ, PODROZE_SEQ i ATRAKCJI PODROZY SEQ.



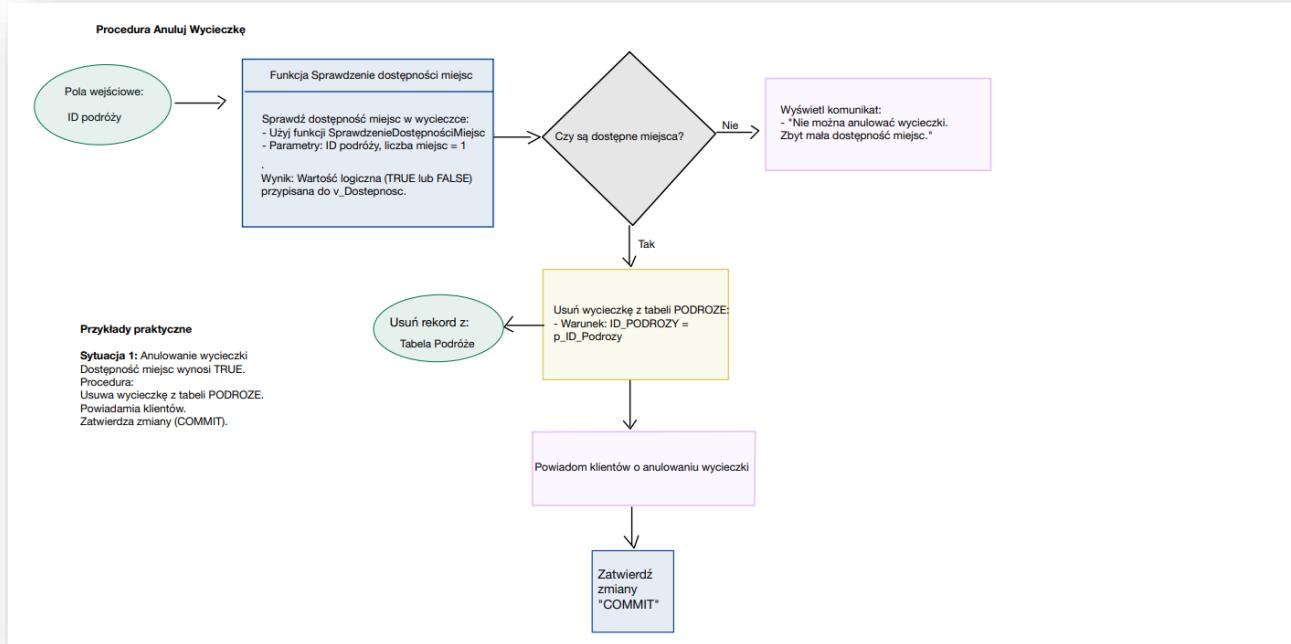
```

1  create or replace PROCEDURE DodajWycieczke (
2      p_Kraj IN VARCHAR2,
3      p_IloscDostepnychMiejsc IN NUMBER,
4      p_IloscZajetychMiejsc IN NUMBER,
5      p_KosztNaOsobe IN FLOAT,
6      p_DataWyjazdu IN DATE,
7      p_DataPrzyjazdu IN DATE,
8      p_ID_Koordynator IN NUMBER,
9      p_NazwaAtrakcji IN VARCHAR2,
10     p_OpisAtrakcji IN VARCHAR2
11 ) IS
12     v_ID_Hotel NUMBER;
13     v_ID_Atrakcji NUMBER;
14     v_ID_Podrozy NUMBER;
15 BEGIN
16     -- Dodaj atrakcję
17     INSERT INTO Atrakcje (
18         ID_ATRAKCJI, NAZWA, OPIS
19     ) VALUES (
20         Atrakcje_seq.NEXTVAL, p_NazwaAtrakcji, p_OpisAtrakcji
21     )
22     RETURNING ID_ATRAKCJI INTO v_ID_Atrakcji;
23
24     -- Przypisz hotel
25     v_ID_Hotel := PrzypiszNowyHotel(NULL, NULL);
26
27     -- Dodaj wycieczkę
28     INSERT INTO Podroze (
29         ID_PODROZY, KRAJ, ILOSC_DOSTEPNYCH_MIEJSC, ILOSC_ZAJETYCH_MIEJSC,
30         KOSZT_NA_OSOBE, DATA_WYJAZDU, DATA_PRZYJAZDU, ID_KOORDYNATOR, ID_HOTEL
31     ) VALUES (
32         Podroze_seq.NEXTVAL, p_Kraj, p_IloscDostepnychMiejsc, p_IloscZajetychMiejsc,
33         p_KosztNaOsobe, p_DataWyjazdu, p_DataPrzyjazdu, p_ID_Koordynator, v_ID_Hotel
34     )
35     RETURNING ID_PODROZY INTO v_ID_Podrozy;
36
37     -- Dodaj wpis do tabeli pośredniej
38     INSERT INTO ATRAKCJI_PODROZY (
39         ID, ATRAKCJE_ID_ATRAKCJI, PODROZE_ID_PODROZY
40     ) VALUES (
41         Atrakcji_Podrozy_seq.NEXTVAL, v_ID_Atrakcji, v_ID_Podrozy
42     );
43
44     COMMIT;
45     DBMS_OUTPUT.PUT_LINE('Wycieczka oraz atrakcja zostały pomyslnie dodane.');
46 EXCEPTION
47     WHEN DUP_VAL_ON_INDEX THEN
48         ROLLBACK;
49         RAISE_APPLICATION_ERROR(-20001, 'Błąd: Duplikat klucza unikalnego.');
50
51     WHEN VALUE_ERROR THEN
52         ROLLBACK;
53         RAISE_APPLICATION_ERROR(-20002, 'Błąd: Nieprawidłowe dane wejściowe.');
54
55     WHEN OTHERS THEN
56         ROLLBACK;
57         RAISE_APPLICATION_ERROR(-20003, 'Błąd podczas dodawania wycieczki i atrakcji: ' || SQLERRM);
58 END DodajWycieczke;
59

```

3. AnulujWycieczke

Cel: Anuluje wycieczkę na podstawie podanego ID podróży.
Logika: Procedura wykorzystuje funkcję SprawdzenieDostępnościMiejsc do sprawdzenia, czy wycieczka może zostać anulowana. Usuwa rekord z tabeli PODROZE i powiadamia klientów. Jeśli anulowanie jest niemożliwe, wyświetla stosowny komunikat.

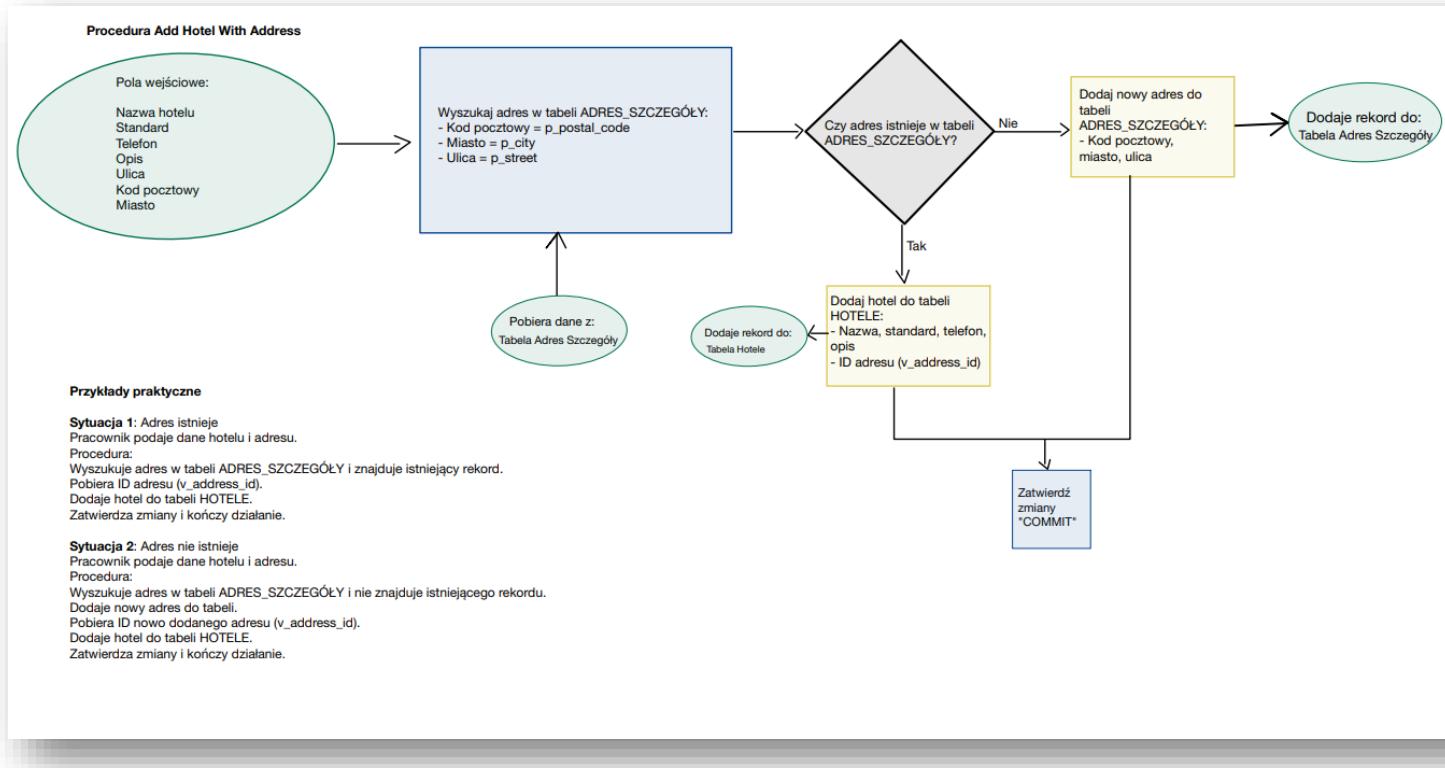


```

1  create or replace PROCEDURE AnulujWycieczke (
2    p_ID_Podrozy IN NUMBER
3  ) IS
4    v_Dostepnosc BOOLEAN; -- Flaga dostępności miejsc
5    v_Koordynator NUMBER; -- ID najgorszego koordynatora
6    BEGIN
7      -- Sprawdź dostępność miejsc (użycie funkcji SprawdzDostepnoscMiejsc)
8      v_Dostepnosc := SprawdzenieDostepnosciMiejsc(p_ID_Podrozy, 1);
9
10     IF v_Dostepnosc THEN
11       -- Znajdź koordynatora z najgorszą efektywnością
12       v_Koordynator := NajgorszyKoordynator();
13
14       DBMS_OUTPUT.PUT_LINE('Koordynator o ID ' || v_Koordynator || |
15       ' ma najgorsze wyniki. Przypisz odpowiedzialność.');
16
17       -- Usuń wycieczkę
18       DELETE FROM Podroze
19       WHERE ID_PODROZY = p_ID_Podrozy;
20
21       -- Powiadom klientów
22       DBMS_OUTPUT.PUT_LINE('Wycieczka została anulowana. Powiadomienie klientów.');
23
24       COMMIT;
25     ELSE
26       DBMS_OUTPUT.PUT_LINE('Nie można anulować wycieczki. Zbyt mała dostępność miejsc.');
27     END IF;
28
29   EXCEPTION
30     WHEN OTHERS THEN
31       ROLLBACK; -- Wycofanie w przypadku błędu
32       DBMS_OUTPUT.PUT_LINE('Błąd podczas anulowania wycieczki: ' || SQLERRM);
33       RAISE;
34   END AnulujWycieczke;
35
  
```

4. ADD_HOTEL_WITH_ADDRESS

Cel: Dodaje nowy hotel wraz z jego adresem do bazy danych.
Logika: Procedura sprawdza, czy adres już istnieje w tabeli ADRES_SZCZEGÓŁY, korzystając z kurSORA c_address. W razie potrzeby dodaje nowy rekord adresu i przypisuje go do nowego hotelu. Wykorzystuje sekwencje ADRES_SEQ i HOTELE_SEQ.



```

1  create or replace PROCEDURE ADD_HOTEL_WITH_ADDRESS (
2      p_hotel_name IN VARCHAR2,
3      p_standard IN VARCHAR2,
4      p_phone IN VARCHAR2,
5      p_description IN VARCHAR2,
6      p_street IN VARCHAR2,
7      p_postal_code IN VARCHAR2,
8      p_city IN VARCHAR2
9  ) IS
10     -- Zmienna na ID adresu
11     v_address_id NUMBER;
12
13     -- Kursor do wyszukiwania adresu
14     CURSOR c_address IS
15         SELECT ID_ADRES
16             FROM ADRES_SZCZEGOLY
17             WHERE KOD_POCZTOWY = p_postal_code
18             AND MIEJSCOWOSC = p_city
19             AND ULICA = p_street;
20
21     BEGIN
22         -- Wyszukiwanie adresu
23         OPEN c_address;
24         FETCH c_address INTO v_address_id;
25
26         IF c_address%NOTFOUND THEN
27             -- Jeśli adres nie istnieje, dodajemy go
28             INSERT INTO ADRES_SZCZEGOLY (ID_ADRES, KOD_POCZTOWY, MIEJSCOWOSC, ULICA)
29             VALUES (ADRES_SEQ.NEXTVAL, p_postal_code, p_city, p_street);
30             v_address_id := ADRES_SEQ.CURRVAL;
31         END IF;
32
33         CLOSE c_address;
34
35         -- Dodawanie hotelu
36         INSERT INTO HOTELE (ID_HOTELU, NAZWA, STANDARD, TELEFON, OPIS, ADRES_SZCZEGOLY_ID_ADRES)
37             VALUES (HOTELE_SEQ.NEXTVAL, p_hotel_name, p_standard, p_phone, p_description, v_address_id);
38
39         COMMIT;
40
41     EXCEPTION
42         WHEN OTHERS THEN
43             ROLLBACK;
44             RAISE_APPLICATION_ERROR(-20002, 'Nieoczekiwany błąd: ' || SQLERRM);
45     END ADD_HOTEL_WITH_ADDRESS;
46

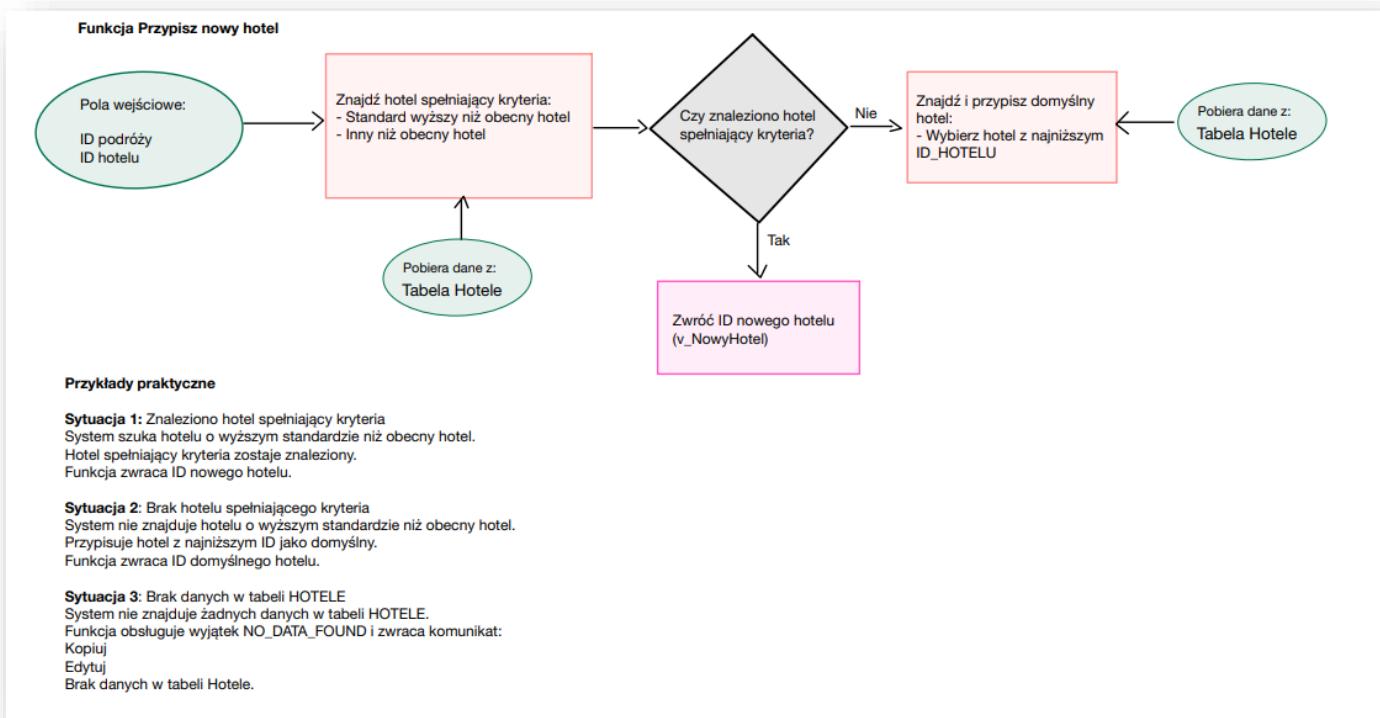
```

Opis funkcji

1. PrzypiszNowyHotel

Cel: Znajduje i przypisuje nowy hotel o wyższym standardzie niż aktualnie przypisany hotel.

Logika: Funkcja zwraca ID nowego hotelu, spełniającego kryteria, lub domyślny hotel, jeśli żaden nie spełnia wymagań.

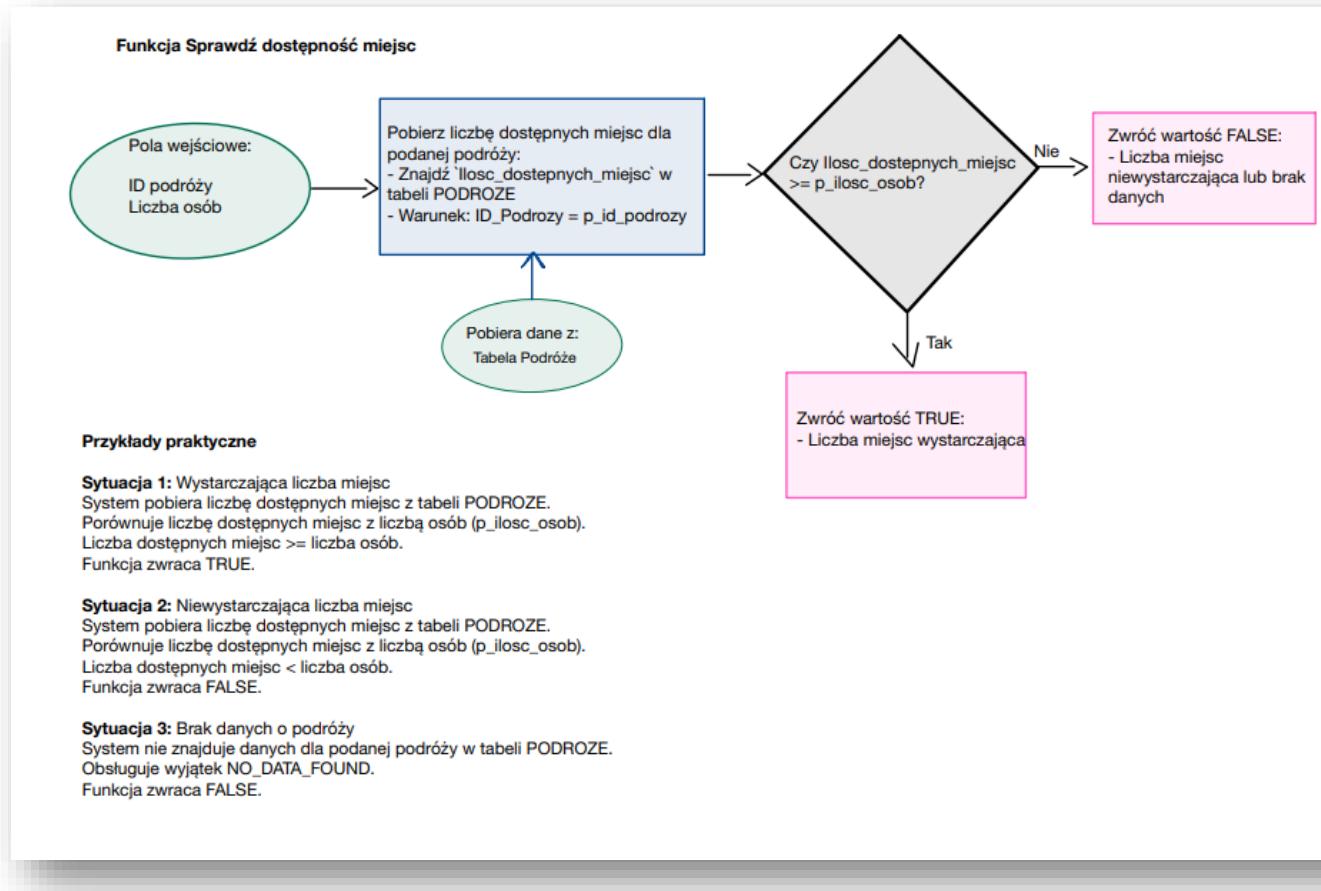


```
1 ✓ create or replace FUNCTION PrzypiszNowyHotel(
2     p_ID_Podrozy IN NUMBER,
3     p_ID_Hotel IN NUMBER
4 ) RETURN NUMBER IS
5     v_NowyHotel NUMBER;
6 ✓ BEGIN
7     -- Logika przypisania nowego hotelu
8     SELECT MIN(ID_HOTELU)
9         INTO v_NowyHotel
10        FROM Hotele
11    WHERE (STANDARD > NVL((SELECT STANDARD FROM Hotele WHERE ID_HOTELU = p_ID_Hotel), 0))
12        AND ID_HOTELU != NVL(p_ID_Hotel, -1);
13
14     -- Jeśli brak nowego hotelu spełniającego kryteria, przypisz domyślny hotel
15    IF v_NowyHotel IS NULL THEN
16        SELECT MIN(ID_HOTELU)
17            INTO v_NowyHotel
18            FROM Hotele;
19    END IF;
20
21    RETURN v_NowyHotel;
22
23 ✓ EXCEPTION
24    WHEN NO_DATA_FOUND THEN
25        RAISE_APPLICATION_ERROR(-20006, 'Brak danych w tabeli Hotele.');
26    WHEN OTHERS THEN
27        RAISE_APPLICATION_ERROR(-20007, 'Błąd w funkcji PrzypiszNowyHotel: ' || SQLERRM);
28 END PrzypiszNowyHotel;
29 /
```

2. SprawdzenieDostępnościMiejsc

Cel: Sprawdza, czy dostępna liczba miejsc w danej podróży wystarcza dla określonej liczby osób.

Logika: Funkcja zwraca wartość TRUE lub FALSE w zależności od dostępności miejsc. Obsługuje przypadki braku danych.



```
1 ✓ create or replace FUNCTION SprawdzenieDostępnościMiejsc(
2     |     p_id_podrozy IN NUMBER,
3     |     p_ilosc_osob IN NUMBER
4   ✓ ) RETURN BOOLEAN IS
5     |     v_dostepnosc INTEGER;
6   ✓ BEGIN
7     |     SELECT Ilosc_dostepnych_miejsc
8     |     INTO v_dostepnosc
9     |     FROM Podroze
10    |     WHERE ID_Podrozy = p_id_podrozy;
11
12   ✓     IF v_dostepnosc >= p_ilosc_osob THEN
13     |     |     RETURN TRUE;
14   ✓     ELSE
15     |     |     RETURN FALSE;
16   ✓     END IF;
17
18   ✓     EXCEPTION
19   ✓     WHEN NO_DATA_FOUND THEN
20     |     |     RETURN FALSE; -- Gdy brak danych o wycieczce
21
22   ✓ END SprawdzenieDostępnościMiejsc;
23 /
```

3. Najgorszy Koordynator

Cel:

Znalezienie koordynatora, który ma najmniejszą sumę sprzedaży, na podstawie kosztów wycieczek i liczby zajętych miejsc.

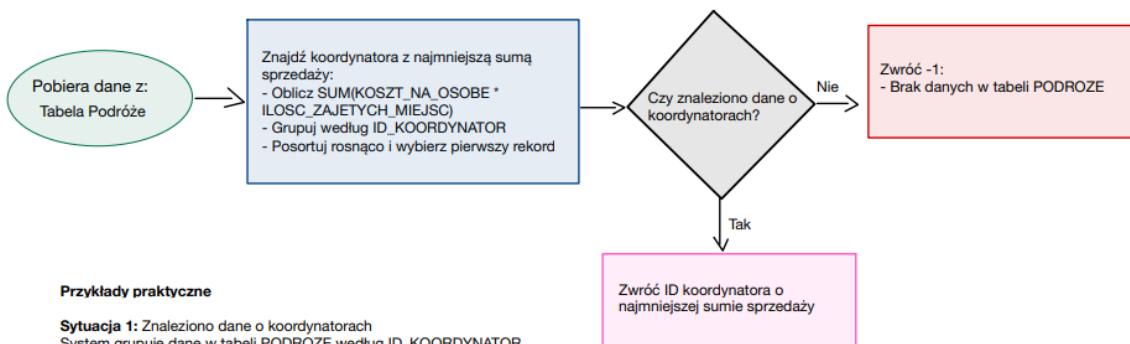
Logika:

Funkcja grupuje dane o wycieczkach według koordynatorów, oblicza ich łączną sprzedaż ($KOSZT_NA_OSOBE * ILOSC_ZAJETYCH_MIEJSC$), a następnie wybiera koordynatora z najmniejszą sumą sprzedaży.

Obsługuje przypadki braku danych i inne błędy, zwracając odpowiednie wartości:

- ID koordynatora: Jeśli dane zostały znalezione.
- -1: Gdy w tabeli PODROZE nie ma żadnych danych.
- -99: Gdy wystąpił inny błąd.

Funkcja Najgorszy Koordynator



Przykłady praktyczne

Sytuacja 1: Znaleziono dane o koordynatorach.
System grupuje dane w tabeli PODROZE według ID_KOORDYNATOR.
Oblicza sumę sprzedaży dla każdego koordynatora ($KOSZT_NA_OSOBE * ILOSC_ZAJETYCH_MIEJSC$).
Znajduje koordynatora z najmniejszą wartością.
Funkcja zwraca ID tego koordynatora.

Sytuacja 2: Brak danych w tabeli.
System nie znajduje żadnych danych w tabeli PODROZE.
Funkcja obsługuje wyjątek NO_DATA_FOUND.
Zwraca wartość -1, sygnalizując brak danych.

```
1  create or replace FUNCTION NajgorszyKoordynator RETURN NUMBER
2  IS
3      v_ID_Koordynator NUMBER;
4  BEGIN
5      -- Znalezienie koordynatora z najmniejszą sumą sprzedaży
6      SELECT ID_KOORDYNATOR
7          INTO v_ID_Koordynator
8          FROM Podroze
9          GROUP BY ID_KOORDYNATOR
10         ORDER BY SUM(KOSZT_NA_OSOBE * ILOSC_ZAJETYCH_MIEJSC) ASC
11         FETCH FIRST 1 ROW ONLY;
12
13     RETURN v_ID_Koordynator;
14 EXCEPTION
15     WHEN NO_DATA_FOUND THEN
16         RETURN -1; -- Brak danych
17     WHEN OTHERS THEN
18         RETURN -99; -- Inny błąd
19 END NajgorszyKoordynator;
20 /
```

Opis sekwencji

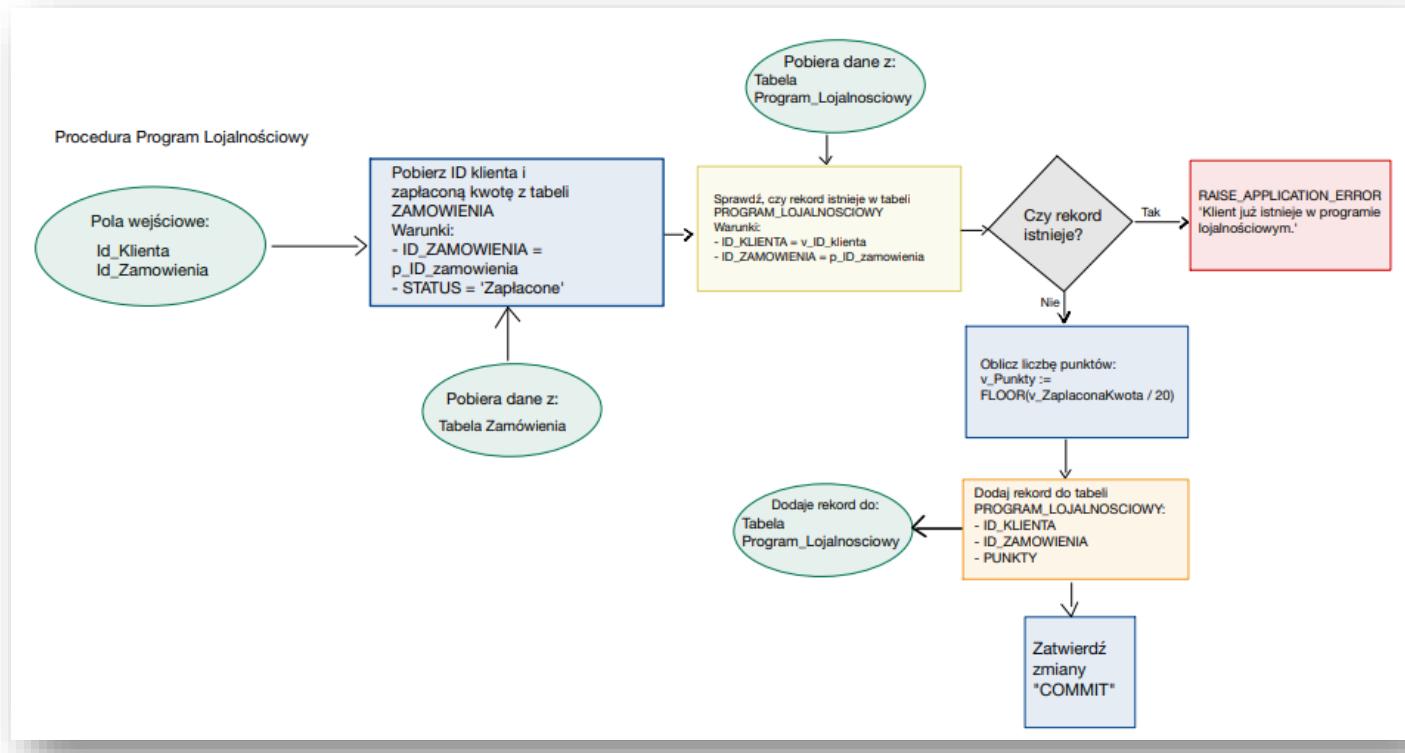
- KOORDYNATOR_SEQ: Generuje unikalne ID dla koordynatorów.
- ATRAKCJE_SEQ: Generuje unikalne ID dla atrakcji.
- PODROZE_SEQ: Generuje unikalne ID dla podróży.
- ATRAKCJI PODROZY_SEQ: Generuje unikalne ID dla relacji między podróżami a atrakcjami.
- ADRES_SEQ: Generuje unikalne ID dla adresów.
- HOTELE_SEQ: Generuje unikalne ID dla hoteli

2.3. Pakiet 3: Analiza i Statystyka

Pakiet nr 3 (Analiza i Statystyki) został zaprojektowany jako uzupełnienie funkcjonalności Pakietu 1 (Klient) oraz Pakietu 2 (Koordynator). Skupia się na zaawansowanej analizie danych i generowaniu statystyk, które wspierają zarówno zarządzanie wycieczkami, jak i ocenę wydajności biura podróży. Dostarcza narzędzi do monitorowania wyników finansowych, analizowania popularności wycieczek, oceny jakości obsługi klientów oraz efektywności pracy koordynatorów. Dzięki temu pozwala podejmować strategiczne decyzje biznesowe i optymalizować ofertę biura podróży.

Procedura PROGRAM_LOJALNOSCIOWY

Procedura umożliwia dodanie klienta do programu lojalnościowego na podstawie opłaconego zamówienia. Pobiera dane o kliencie i kwocie zapłaconej z tabeli ZAMÓWIENIA, sprawdza, czy klient już znajduje się w programie lojalnościowym, a następnie oblicza liczbę punktów na podstawie zapłaconej kwoty (1 punkt za każde 20 zł). W przypadku spełnienia warunków dodaje klienta do tabeli PROGRAM_LOJALNOSCIOWY i zatwierdza zmiany w bazie danych za pomocą COMMIT. W razie próby dodania klienta, który już istnieje w programie, procedura generuje błąd.



```

1  create or replace PROCEDURE PrzypnajPunktyLojalnosciowe (
2      p_ID_zamowienia IN NUMBER
3  ) AS
4      v_ID_klienta NUMBER;
5      v_ZaplaconaKwota NUMBER;
6      v_Punkty NUMBER;
7      v_RekordIstnieje NUMBER;
8  BEGIN
9      -- Pobierz ID klienta oraz zapłaconą kwotę z tabeli ZAMOWIENIA
10     SELECT KLIENCI_ID_Klienta, KWOTA
11     INTO v_ID_klienta, v_ZaplaconaKwota
12     FROM ZAMOWIENIA
13     WHERE ID_ZAMOWIENIA = p_ID_zamowienia
14     AND STATUS = 'Zapłacone';
15
16     -- Sprawdź, czy rekord dla tego klienta już istnieje
17     SELECT COUNT(*)
18     INTO v_RekordIstnieje
19     FROM PROGRAM_LOJALNOSCIOVY
20     WHERE ID_Klienta = v_ID_klienta
21     AND ID_ZAMOWIENIA = p_ID_zamowienia;
22
23     IF v_RekordIstnieje > 0 THEN
24         -- Jeżeli rekord istnieje, zgłoś błąd
25         RAISE_APPLICATION_ERROR(-20003, 'Klient już istnieje w programie lojalnościowym.');
26     ELSE
27         -- Oblicz liczbę punktów (np. 1 punkt za każde 20 zł)
28         v_Punkty := FLOOR(v_ZaplaconaKwota / 20);
29
30         -- Dodaj nowy wpis do tabeli PROGRAM_LOJALNOSCIOVY
31         INSERT INTO PROGRAM_LOJALNOSCIOVY (ID_Klienta, ID_ZAMOWIENIA, PUNKTY)
32         VALUES (v_ID_klienta, p_ID_zamowienia, v_Punkty);
33     END IF;
34
35     -- Zatwierdź zmiany
36     COMMIT;
37 EXCEPTION
38     WHEN NO_DATA_FOUND THEN
39         RAISE_APPLICATION_ERROR(-20001, 'Nie znaleziono zamówienia o podanym ID lub zamówienie nie jest opłacone.');
40     WHEN OTHERS THEN
41         RAISE_APPLICATION_ERROR(-20002, 'Wystąpił błąd: ' || SQLERRM);
42 END PrzypnajPunktyLojalnosciowe;
43

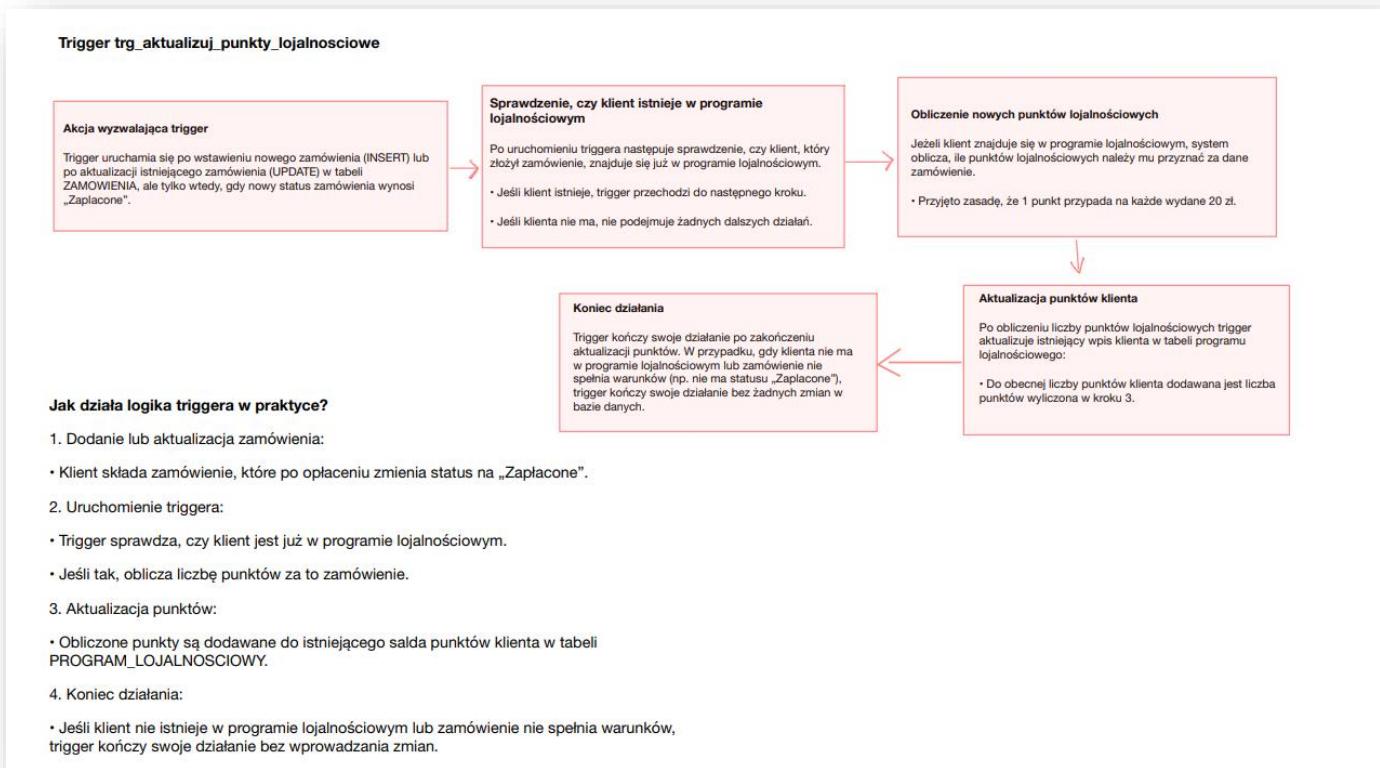
```

Trigger trg_aktualizuj_punkty_lojalnosciowe

Trigger automatycznie aktualizuje punkty lojalnościowe klienta po opłaceniu zamówienia. Uruchamia się po wstawieniu nowego zamówienia (INSERT) lub aktualizacji statusu zamówienia na Zapłacone.

Działanie:

- Sprawdza, czy klient znajduje się już w programie lojalnościowym.
- Jeśli tak, oblicza liczbę punktów na podstawie kwoty zamówienia (1 punkt za każde 20 zł) i dodaje je do istniejącej liczby punktów w tabeli PROGRAM_LOJALNOŚCIOWY.
- W przypadku braku klienta w programie kończy działanie bez wprowadzania zmian.



```

1  create or replace TRIGGER trg_aktualizuj_punkty_lojalnosciowe
2  AFTER INSERT OR UPDATE ON ZAMOWIENIA
3  FOR EACH ROW
4  WHEN (NEW.STATUS = 'Zaplacone') -- Działa tylko dla opłaconych zamówień
5  DECLARE
6      v_Licznik NUMBER;
7      v_NowePunkty NUMBER;
8  BEGIN
9      -- Sprawdź, czy klient już istnieje w programie lojalnościowym
10     SELECT COUNT(*)
11     INTO v_Licznik
12     FROM PROGRAM_LOJALNOSCIOWY
13     WHERE ID_Klienta = :NEW.KLIENCI_ID_Klienta;
14
15     IF v_Licznik > 0 THEN
16         -- Oblicz punkty dla nowego zamówienia (1 punkt za każde 20 zł)
17         v_NowePunkty := FLOOR(:NEW.KWOTA / 20);
18
19         -- Zaktualizuj punkty klienta w programie lojalnościowym
20         UPDATE PROGRAM_LOJALNOSCIOWY
21         SET PUNKTY = PUNKTY + v_NowePunkty
22         WHERE ID_Klienta = :NEW.KLIENCI_ID_Klienta;
23     ELSE
24         -- Jeśli klient jeszcze nie istnieje, nie rób nic (opcjonalnie można dodać logikę)
25         NULL;
26     END IF;
27 END trg_aktualizuj_punkty_lojalnosciowe;
28 /

```

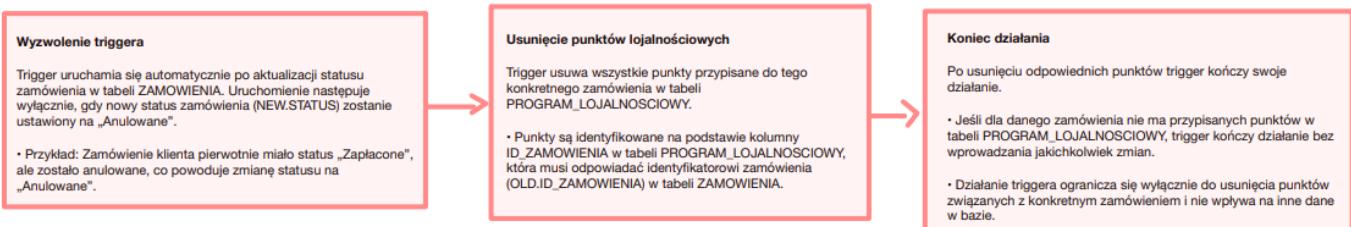
Trigger trg_usun_punkty_anulowane

Trigger usuwa punkty lojalnościowe przypisane do zamówienia, które zmieniło status na Anulowane. Uruchamia się automatycznie po aktualizacji statusu zamówienia.

Działanie:

- Sprawdza, czy dane zamówienie ma przypisane punkty w tabeli PROGRAM_LOJALNOŚCIOWY.
- Usuwa punkty powiązane z anulowanym zamówieniem.
- Jeśli nie znaleziono punktów do usunięcia, kończy działanie bez wprowadzania zmian.

Trigger trg_usun_punkty_anulowane



Jak działa logika triggera w praktyce?

1. Aktualizacja zamówienia:

- Pracownik lub system zmienia status zamówienia na „Anulowane”.

2. Uruchomienie triggera:

- Trigger aktywuje się, sprawdza, czy nowy status zamówienia to „Anulowane”, i przechodzi do kolejnych kroków.

3. Usunięcie punktów:

- Jeśli zamówienie posiadało przypisane punkty lojalnościowe w tabeli PROGRAM_LOJALNOSCIOVY, punkty te zostają usunięte na podstawie identyfikatora zamówienia.

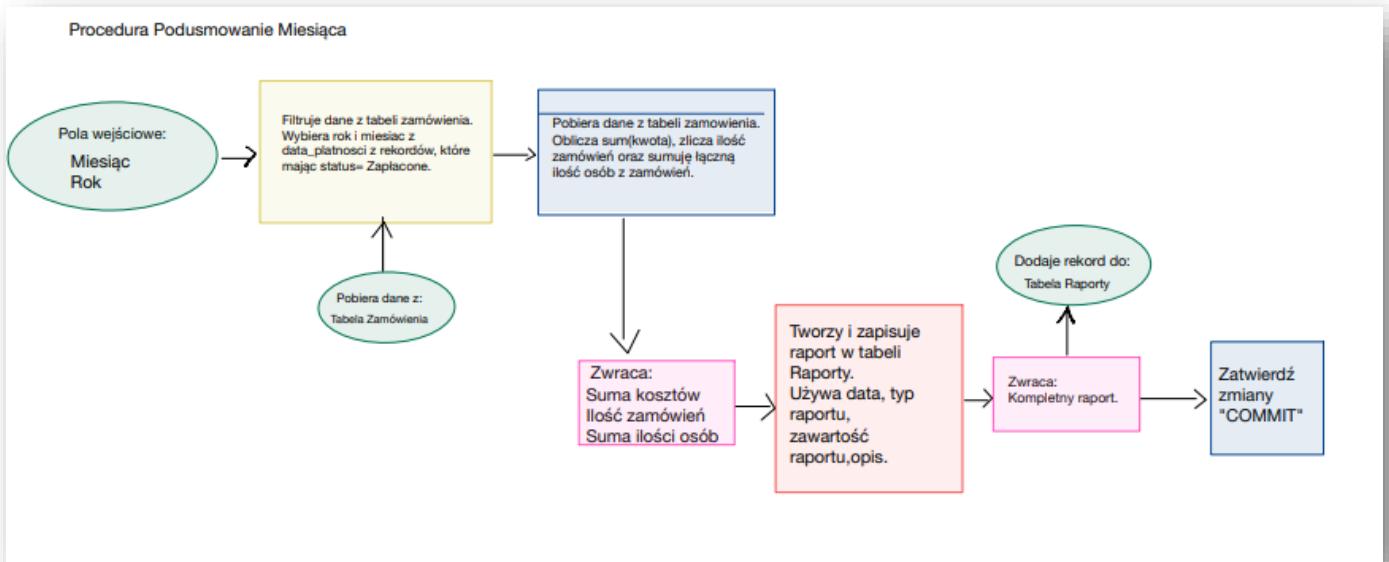
4. Koniec działania:

- Jeśli zamówienie nie posiadało żadnych punktów lub status zmiany nie spełniał warunków (np. inny niż „Anulowane”), trigger kończy swoje działanie bez wprowadzania zmian.

```
1  create or replace TRIGGER trg_usun_punkty_anulowane
2  AFTER UPDATE OF STATUS ON ZAMOWIENIA
3  FOR EACH ROW
4  WHEN (NEW.STATUS = 'Anulowane') -- Działa tylko, gdy nowy status to "Anulowane"
5  BEGIN
6      -- Usuń punkty tylko dla tego zamówienia
7      DELETE FROM PROGRAM_LOJALNOSCIOVY
8      WHERE ID_ZAMOWIENIA = :OLD.ID_ZAMOWIENIA;
9  END;
10 /
```

Procedura PODSUMOWANIE_MIESIĄCA

Procedura umożliwia stworzenie raportu podsumowującego dane finansowe i operacyjne dla wskazanego miesiąca i roku. Filtruje dane z tabeli ZAMÓWIENIA, uwzględniając tylko zamówienia o statusie „Zapłacone”, oblicza sumę kosztów, liczbę zamówień oraz sumę liczby osób. Następnie tworzy raport z podsumowaniem tych danych, zapisuje go w tabeli RAPORTY i zatwierdza zmiany w bazie danych.



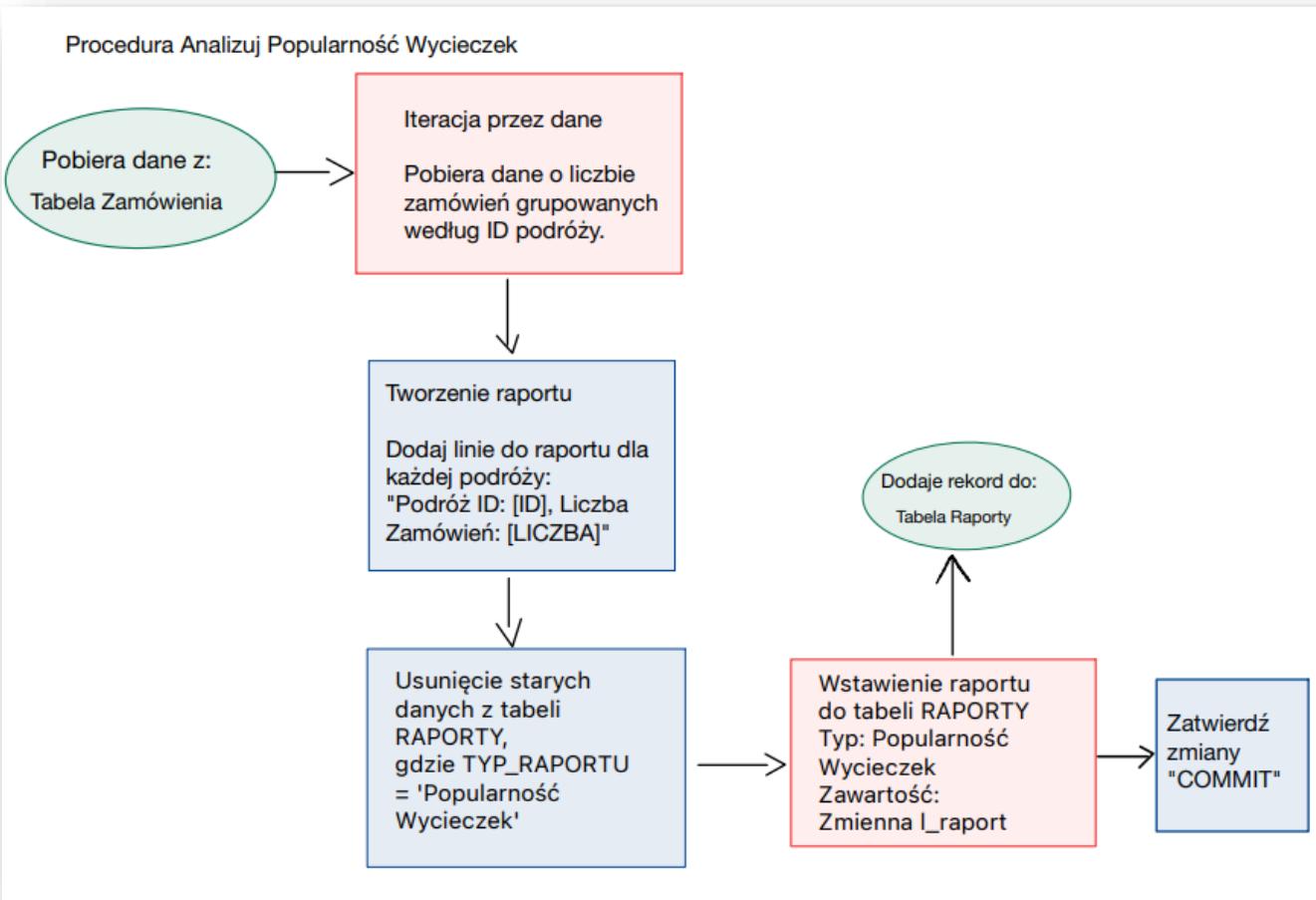
```

1  create or replace PROCEDURE PODSUMOWANIE_MIESIACA (
2      p_rok IN NUMBER,
3      p_miesiac IN NUMBER
4  ) IS
5      v_przychod NUMBER := 0;
6      v_liczba_zamowien NUMBER := 0;
7      v_ilosc_osob NUMBER := 0;
8  BEGIN
9      -- Pobierz dane podsumowania z tabeli ZAMOWIENIA
10     SELECT
11         NVL(SUM(kwota), 0),          -- Suma przychodów
12         NVL(COUNT(*), 0),           -- Liczba zamówień
13         NVL(SUM(ilosc_osob), 0)    -- Suma osób
14     INTO
15         v_przychod,
16         v_liczba_zamowien,
17         v_ilosc_osob
18     FROM ZAMOWIENIA
19     WHERE EXTRACT(YEAR FROM data_platnosci) = p_rok
20         AND EXTRACT(MONTH FROM data_platnosci) = p_miesiac
21         AND status = 'Zapłacone';
22
23     -- Wstaw dane do tabeli RAPORTY
24     INSERT INTO RAPORTY (
25         DATA,
26         TYP_RAPORTU,
27         ZAWARTOSC_RAPORTU,
28         OPIS
29     ) VALUES (
30         SYSDATE,
31         'Podsumowanie miesiąca',
32         'Podsumowanie dla miesiąca: ' || p_miesiac || '/' || p_rok ||
33         ' Przychód: ' || v_przychod ||
34         ' PLN Liczba zamówień: ' || v_liczba_zamowien ||
35         ' Łączna liczba osób: ' || v_ilosc_osob,
36         'Raport automatyczny wygenerowany przez procedure'
37     );
38
39     COMMIT;
40 END;
41

```

Procedura ANALIZUJ_POPULARNOŚĆ_WYCIECZEK

Procedura umożliwia analizę popularności wycieczek. Pobiera dane z tabeli ZAMÓWIENIA, oblicza liczbę zamówień dla każdej podróży, tworzy raport zawierający informacje o liczbie zamówień dla poszczególnych wycieczek, usuwa stare dane z tabeli RAPORTY i zapisuje nowy raport w tej tabeli. Na końcu procedura zatwierdza zmiany w bazie danych.



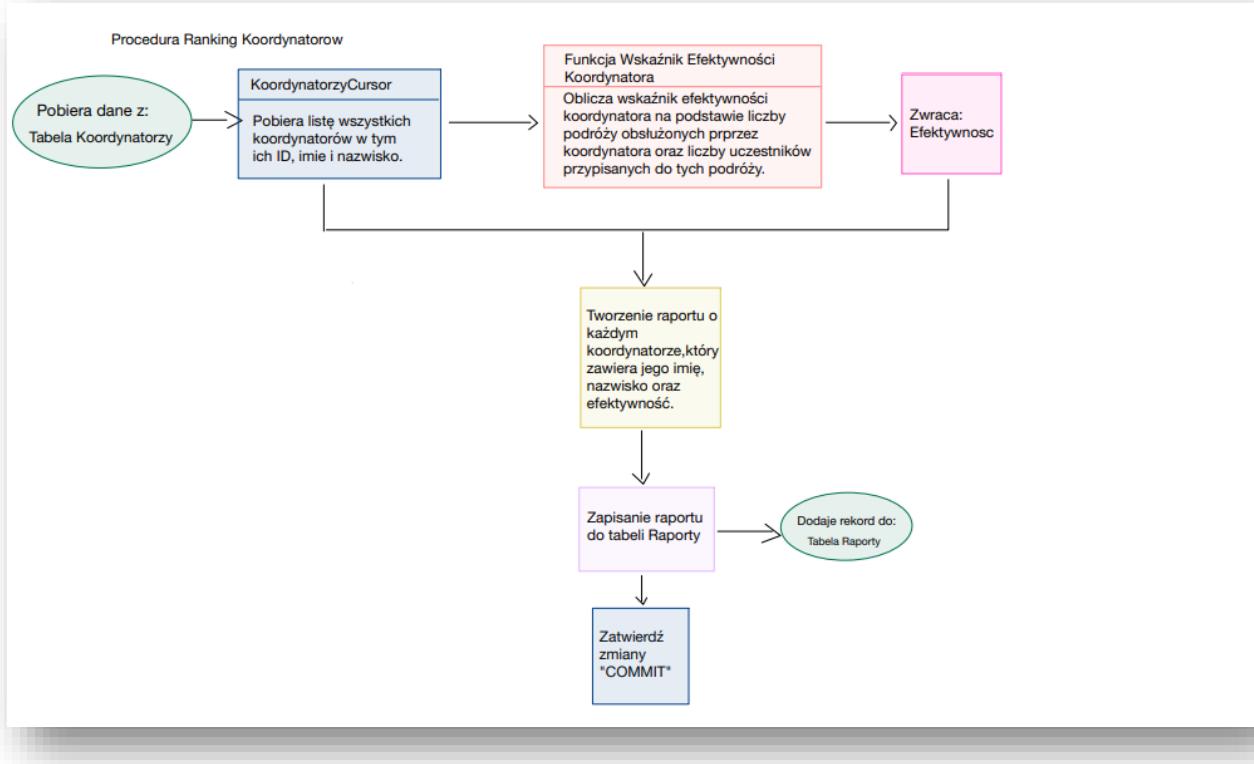
```

1  create or replace PROCEDURE AnalizujPopularoscWycieczek
2  IS
3      -- Deklaracja zmiennych
4      l_raport CLOB := EMPTY_CLOB(); -- Bufor do przechowywania raportu
5      l_first BOOLEAN := TRUE;          -- Flaga do zarządzania przecinkami
6  BEGIN
7      -- Iteracja przez dane i budowa raportu
8      FOR rec IN (SELECT PODROZE_ID_PODROZY, COUNT(*) AS LICZBA_ZAMOWIEN
9                  FROM ZAMOWIENIA
10                 GROUP BY PODROZE_ID_PODROZY
11                ORDER BY LICZBA_ZAMOWIEN DESC) LOOP
12          -- Dodanie nowej linii do raportu
13          l_raport := l_raport || 'Podróz ID: ' || rec.PODROZE_ID_PODROZY || ', Liczba Zamówień: ' || rec.LICZBA_ZAMOWIEN || CHR(10);
14      END LOOP;
15
16      -- Usunięcie starych danych raportu tego typu
17      DELETE FROM RAPORTY WHERE TYP_RAPORTU = 'Popularność Wycieczek';
18
19      -- Wstawienie całego raportu jako jeden rekord
20      INSERT INTO RAPORTY (TYP_RAPORTU, DATA, ZAWARTOSC_RAPORTU, OPIS)
21      VALUES ('Popularność Wycieczek',
22              SYSDATE,
23              l_raport,
24              'Raport popularności wycieczek');
25
26      -- Zatwierdzenie zmian
27      COMMIT;
28
29  EXCEPTION
30      WHEN OTHERS THEN
31          -- Obsługa błędów
32          ROLLBACK;
33          DBMS_OUTPUT.PUT_LINE('Wystąpił błąd: ' || SQLERRM);
34  END AnalizujPopularoscWycieczek;
35

```

Procedura RANKING_KOORDYNATORÓW

Procedura umożliwia stworzenie rankingu koordynatorów na podstawie ich efektywności. Pobiera listę wszystkich koordynatorów, oblicza wskaźnik efektywności dla każdego z nich, tworzy raport zawierający imię, nazwisko i obliczoną efektywność, a następnie zapisuje ten raport w tabeli RAPORTY. Procedura kończy się zatwierdzeniem zmian w bazie danych za pomocą COMMIT.



```

1  create or replace PROCEDURE RankingKoordynatorow IS
2      -- Zmienna do przechowywania wyników zapytania
3      CURSOR KoordynatorzyCursor IS
4          SELECT ID_Koordynator, Imie, Nazwisko
5              FROM KOORDYNATOR;
6
7      v_ID_Koordynator KOORDYNATOR.ID_Koordynator%TYPE;
8      v_Imie KOORDYNATOR.Imie%TYPE;
9      v_Nazwisko KOORDYNATOR.Nazwisko%TYPE;
10     v_Efektywnosc NUMBER;
11
12     -- Raport w formacie CLOB
13     v_Report CLOB := EMPTY_CLOB();
14
15 BEGIN
16     -- Otwórz cursor
17     OPEN KoordynatorzyCursor;
18
19     LOOP
20         FETCH KoordynatorzyCursor INTO v_ID_Koordynator, v_Imie, v_Nazwisko;
21         EXIT WHEN KoordynatorzyCursor%NOTFOUND;
22
23         -- Wywołanie funkcji do obliczenia efektywności koordynatora
24         v_Efektywnosc := WskaznikEfektywnosciKoordynatora(v_ID_Koordynator);
25
26         -- Dodanie informacji o koordynatorze do raportu
27         v_Report := v_Report || 'Koordynator: ' || v_Imie || ' ' || v_Nazwisko ||
28             ', Efektywnosc: ' || TO_CHAR(v_Efektywnosc) || CHR(10);
29     END LOOP;
30
31     -- Zamknij cursor
32     CLOSE KoordynatorzyCursor;
33
34     -- Zapisanie raportu do tabeli RAPORTY
35     INSERT INTO RAPORTY (DATA, TYP_RAPORTU, ZAWARTOSC_RAPORTU, OPIS)
36     VALUES (SYSDATE, 'Ranking koordynatorów', v_Report, 'Ranking koordynatorów na podstawie efektywności');
37
38     -- Zatwierdzenie transakcji
39     COMMIT;
40 END RankingKoordynatorow;
41

```

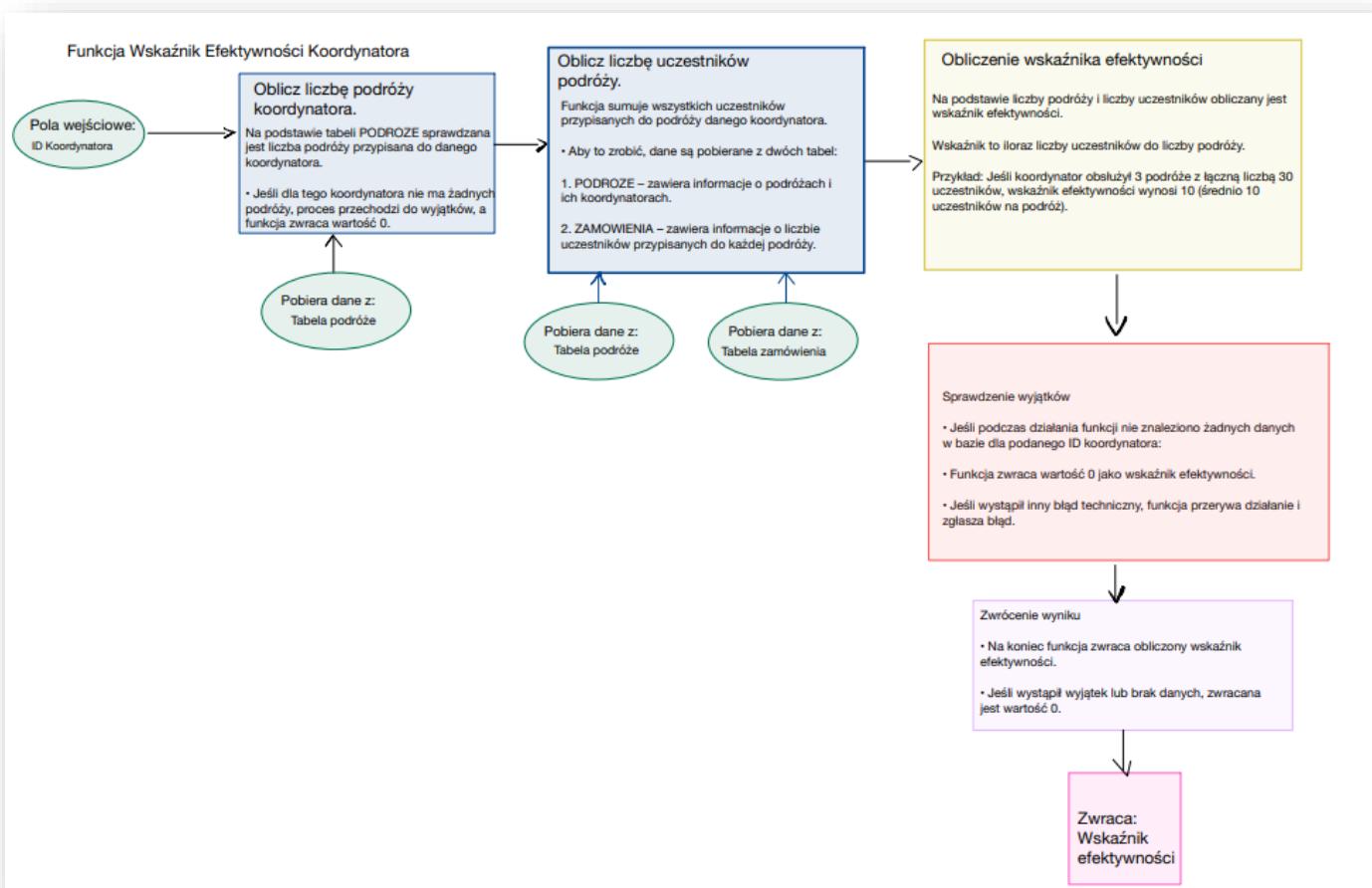
Funkcja oblicza wskaźnik efektywności koordynatora na podstawie liczby podróży przez niego obsłużonych oraz liczby uczestników przypisanych do tych podróży.

- **Pola wejściowe:** ID Koordynatora.

- **Kroki działania:**

1. Oblicza liczbę podróży przypisanych do koordynatora z tabeli PODRÓŻE.
2. Sumuje liczbę uczestników przypisanych do podróży koordynatora, wykorzystując dane z tabel PODRÓŻE i ZAMÓWIENIA.
3. Na podstawie obliczonych danych oblicza wskaźnik efektywności:
Wskaźnik efektywności = Liczba uczestników/Liczba podróży.
4. Obsługuje wyjątki, np. brak danych lub błędy techniczne, zwracając odpowiednie wartości (np. 0 dla braku danych).
5. Zwraca obliczony wskaźnik efektywności.

Funkcja jest kluczowa dla działania procedury RANKING_KOORDYNATORÓW, zapewniając dokładne dane do tworzenia raportu.



```

1  create or replace FUNCTION WskaznikEfekt Untitled-1
2      p_ID_Koordynator IN NUMBER
3  ) RETURN NUMBER IS
4      v_LiczbaPodrozy NUMBER;
5      v_LiczbaUczestnikow NUMBER;
6      v_WskaznikEfektywnosci NUMBER := 0;
7  BEGIN
8      -- Oblicz liczbę podróży koordynatora
9      SELECT COUNT(*)
10     INTO v_LiczbaPodrozy
11     FROM PODROZE
12     WHERE ID_KOORDYNATOR = p_ID_Koordynator;
13
14      -- Oblicz liczbę uczestników w tych podróżach
15      SELECT SUM(Z.ILOSC_OSOB)
16     INTO v_LiczbaUczestnikow
17     FROM ZAMOWIENIA Z
18     JOIN PODROZE P ON Z.PODROZE_ID_PODROZY = P.ID_PODROZY
19     WHERE P.ID_KOORDYNATOR = p_ID_Koordynator;
20
21      -- Oblicz wskaźnik efektywności
22      IF v_LiczbaPodrozy > 0 THEN
23          v_WskaznikEfektywnosci := v_LiczbaUczestnikow / v_LiczbaPodrozy;
24      END IF;
25
26      RETURN v_WskaznikEfektywnosci;
27  EXCEPTION
28      WHEN NO_DATA_FOUND THEN
29          -- Jeśli brak danych, zwróć 0
30          RETURN 0;
31      WHEN OTHERS THEN
32          -- W przypadku innych błędów, zgłoś wyjątek
33          RAISE;
34  END WskaznikEfektywnosciKoordynatora;
35

```

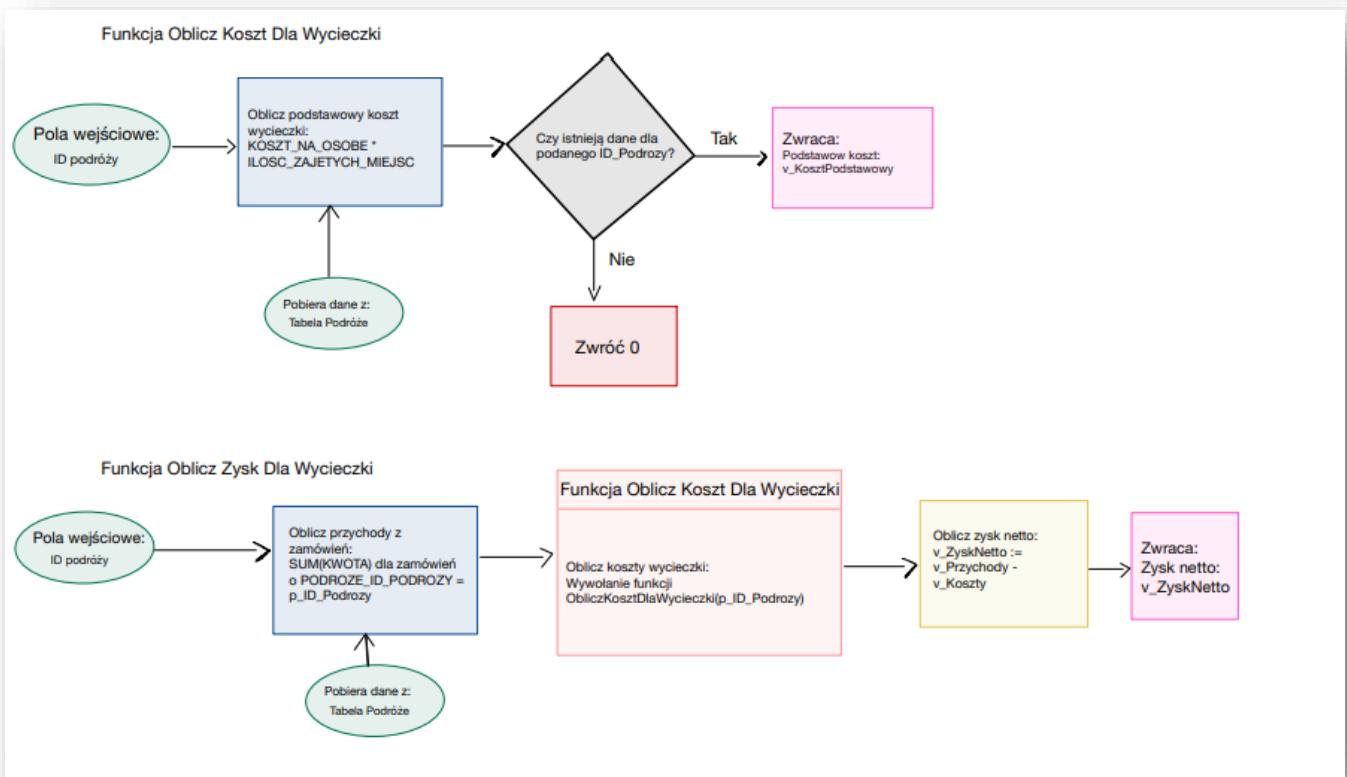
Funkcja OBLICZ_KOSZT_DLA_WYCIECZKI

Funkcja umożliwia obliczenie podstawowego kosztu wycieczki na podstawie danych dotyczących liczby zajętych miejsc oraz kosztu na osobę. Sprawdza, czy istnieją dane dla podanego ID_PODRÓŻY. W przypadku braku danych zwraca wartość 0, a w przypadku poprawnych danych zwraca obliczony koszt podstawowy.

Funkcja OBLICZ_ZYSK_DLA_WYCIECZKI

Funkcja umożliwia obliczenie zysku netto dla wycieczki na podstawie danych dotyczących przychodów z zamówień oraz kosztów wycieczki. Najpierw oblicza całkowite przychody, sumując kwoty zamówień dla podanego ID_PODRÓŻY. Następnie wywołuje funkcję OBLICZ_KOSZT_DLA_WYCIECZKI, aby uzyskać koszty podstawowe wycieczki. Na tej podstawie funkcja oblicza różnicę między przychodami a kosztami, co daje wartość zysku netto. W

przypadku braku danych lub błędów zwraca wartość 0. Funkcja pozwala na analizę opłacalności wycieczek i może być wykorzystana w procesach raportowania finansowego.



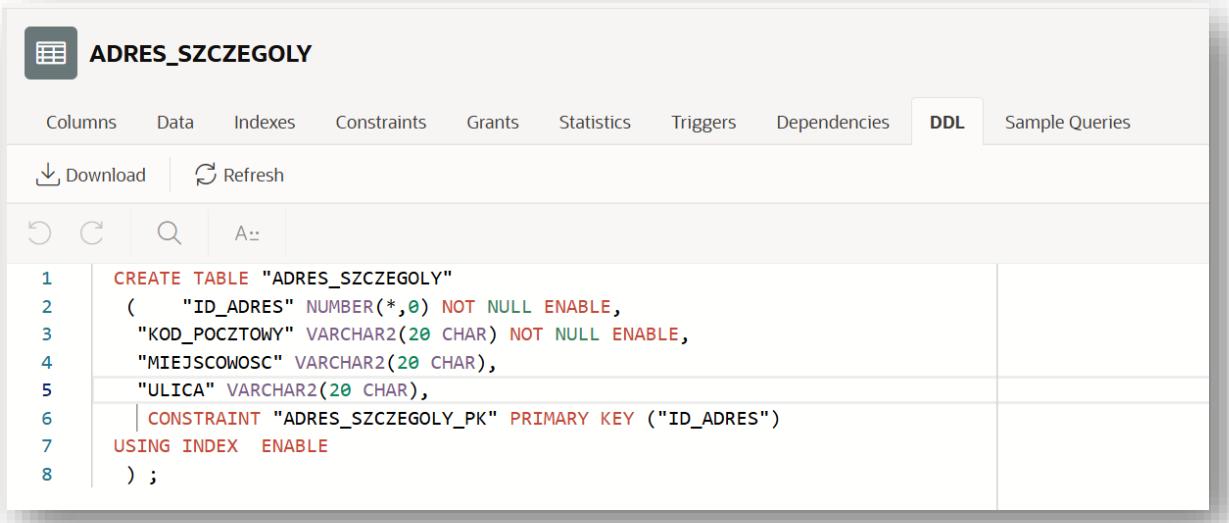
```

1  create or replace FUNCTION OBLICZ_KOSZT_REZERWACJI (
2      p_id_podrozy IN NUMBER,
3      p_ilosc_osob IN NUMBER
4  ) RETURN NUMBER IS
5      v_koszt_na_osobe NUMBER;
6  BEGIN
7      SELECT KOSZT_NA_OSÓBE
8          INTO v_koszt_na_osobe
9          FROM PODROZE
10         WHERE ID_PODROZY = p_id_podrozy;
11
12     RETURN v_koszt_na_osobe * p_ilosc_osob;
13 END;
14 /
  
```

```
1 ∵ create or replace FUNCTION ObliczZyskDlaWycieczki (
2   | p_ID_Podrozy IN NUMBER
3 ∵ ) RETURN NUMBER IS
4   | v_Przychody NUMBER := 0;
5   | v_Koszty NUMBER := 0;
6   | v_ZyskNetto NUMBER := 0;
7 ∵ BEGIN
8   -- Oblicz przychody z zamówień dla wycieczki
9   SELECT NVL(SUM(KWOTA), 0)
10  INTO v_Przychody
11  FROM ZAMOWIENIA
12  WHERE PODROZE_ID_PODROZY = p_ID_Podrozy;
13
14  -- Oblicz koszty wycieczki (wykorzystanie funkcji ObliczKosztDlaWycieczki)
15  v_Koszty := ObliczKosztDlaWycieczki(p_ID_Podrozy);
16
17  -- Oblicz zysk netto
18  v_ZyskNetto := v_Przychody - v_Koszty;
19
20  -- Zwróć zysk netto
21  RETURN v_ZyskNetto;
22
23 ∵ EXCEPTION
24 ∵   WHEN NO_DATA_FOUND THEN
25   |   RETURN 0; -- Jeśli brak danych, zwróć 0
26 ∵   WHEN OTHERS THEN
27   |   RETURN 0; -- Obsługa innych błędów
28 END ObliczZyskDlaWycieczki;
29 /
```

3. Apex

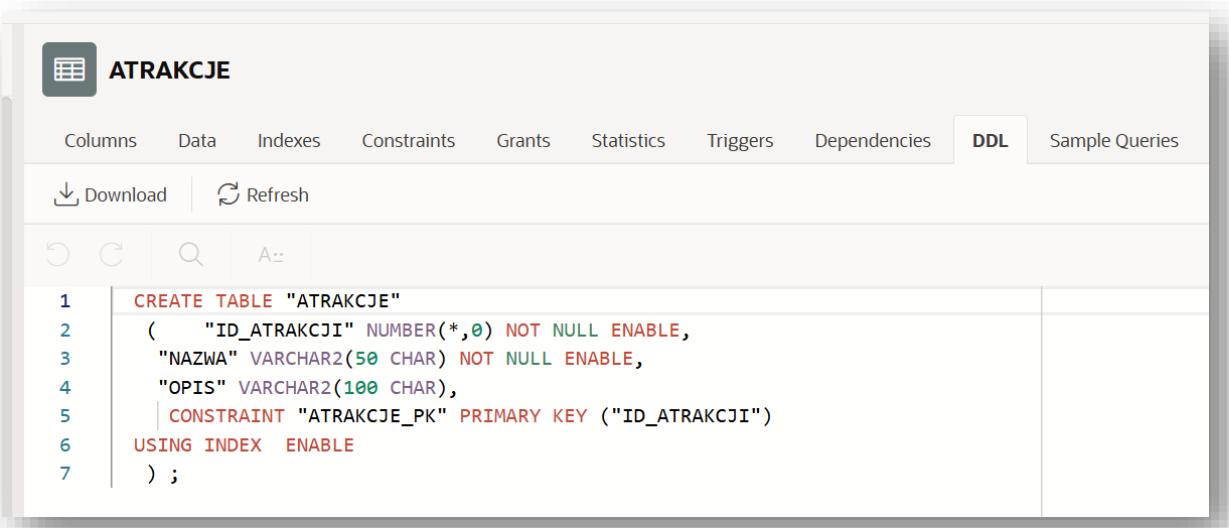
3.1. Tworzenie tabel



The screenshot shows the Oracle SQL Developer interface for the table "ADRES_SZCZEGOLY". The "DDL" tab is selected. The code listed is:

```
1 CREATE TABLE "ADRES_SZCZEGOLY"
2   (
3     "ID_ADRES" NUMBER(*,0) NOT NULL ENABLE,
4     "KOD_POCZTOWY" VARCHAR2(20 CHAR) NOT NULL ENABLE,
5     "MIEJSCOWOSC" VARCHAR2(20 CHAR),
6     "ULICA" VARCHAR2(20 CHAR),
7     | CONSTRAINT "ADRES_SZCZEGOLY_PK" PRIMARY KEY ("ID_ADRES")
8   USING INDEX ENABLE
9   ) ;
```

Tabela 1 Adres Szczegóły



The screenshot shows the Oracle SQL Developer interface for the table "ATRAKCJE". The "DDL" tab is selected. The code listed is:

```
1 CREATE TABLE "ATRAKCJE"
2   (
3     "ID_ATRAKCJI" NUMBER(*,0) NOT NULL ENABLE,
4     "NAZWA" VARCHAR2(50 CHAR) NOT NULL ENABLE,
5     "OPIS" VARCHAR2(100 CHAR),
6     | CONSTRAINT "ATRAKCJE_PK" PRIMARY KEY ("ID_ATRAKCJI")
7   USING INDEX ENABLE
8   ) ;
```

Tabela 2 Atrakcje

ATRAKCJI_PODROZY

Columns Data Indexes Constraints Grants Statistics Triggers Dependencies **DDL** Sample Queries

Download Refresh

A: A..

```

8
9   ALTER TABLE "ATRAKCJI_PODROZY" ADD CONSTRAINT "ATRAKCJI_PODROZY_ATRAKCJE_FK" FOREIGN KEY ("ATRAKCJE_ID_ATRAKCJI")
10  REFERENCES "ATRAKCJE" ("ID_ATRAKCJI") ENABLE;
11  ALTER TABLE "ATRAKCJI_PODROZY" ADD CONSTRAINT "ATRAKCJI_PODROZY_PODROZE_FK" FOREIGN KEY ("PODROZE_ID_PODROZY")
12  REFERENCES "PODROZE" ("ID_PODROZY") ON DELETE CASCADE ENABLE;
13

```

Tabela 3 Atrakcji_Podrozy

HOTELE

Columns Data Indexes Constraints Grants Statistics Triggers Dependencies **DDL** Sample Queries

Download Refresh

A: A..

```

1 CREATE TABLE "HOTELE"
2   (
3     "ID_HOTELU" NUMBER(*,0) NOT NULL ENABLE,
4     "NAZWA" VARCHAR2(20 CHAR) NOT NULL ENABLE,
5     "STANDARD" VARCHAR2(20 CHAR) NOT NULL ENABLE,
6     "TELEFON" VARCHAR2(12 CHAR) NOT NULL ENABLE,
7     "OPIS" VARCHAR2(100) NOT NULL ENABLE,
8     "ADRES_SZCZEGOLY_ID_ADRES" NUMBER(*,0) NOT NULL ENABLE,
9     | CONSTRAINT "HOTELE_PK" PRIMARY KEY ("ID_HOTELU")
10    USING INDEX ENABLE
11  );
12
13  ALTER TABLE "HOTELE" ADD CONSTRAINT "HOTELE_ADRES_SZCZEGOLY_FK" FOREIGN KEY ("ADRES_SZCZEGOLY_ID_ADRES")
14    REFERENCES "ADRES_SZCZEGOLY" ("ID_ADRES") ENABLE;

```

Tabela 4 Hotele

KLIENCI

Columns Data Indexes Constraints Grants Statistics Triggers Dependencies **DDL** Sample Queries

Download Refresh

↻ ↺ ⌂ A..

```

1 CREATE TABLE "KLIENCI"
2   ( "ID_Klienta" NUMBER(*,0) NOT NULL ENABLE,
3     "NAZWISKO" VARCHAR2(20 CHAR) NOT NULL ENABLE,
4     "IMIE" VARCHAR2(20 CHAR) NOT NULL ENABLE,
5     "NIP" VARCHAR2(12 CHAR),
6     "NAZWA_FIRMY" VARCHAR2(20 CHAR),
7     "ADRES" NUMBER(*,0) NOT NULL ENABLE,
8     "EMAIL" VARCHAR2(50) NOT NULL ENABLE,
9     "TELEFON" VARCHAR2(12 CHAR) NOT NULL ENABLE,
10    | CONSTRAINT "KLIENCI_PK" PRIMARY KEY ("ID_Klienta")
11    USING INDEX ENABLE
12  );
13
14 ALTER TABLE "KLIENCI" ADD CONSTRAINT "KLIENCI_ADRES_SZCZEGOLY_FK" FOREIGN KEY ("ADRES")
15   REFERENCES "ADRES_SZCZEGOLY" ("ID_Adres") ENABLE;

```

Tabela 5 Klienci

KOORDYNATOR

Columns Data Indexes Constraints Grants Statistics Triggers Dependencies **DDL** Sample Queries

Download Refresh

↻ ↺ ⌂ A..

```

1 CREATE TABLE "KOORDYNATOR"
2   ( "ID_Koordinatatora" NUMBER(*,0) NOT NULL ENABLE,
3     "NAZWISKO" VARCHAR2(20 CHAR) NOT NULL ENABLE,
4     "IMIE" VARCHAR2(20 CHAR) NOT NULL ENABLE,
5     "EMAIL" VARCHAR2(50 CHAR) NOT NULL ENABLE,
6     "TELEFON" VARCHAR2(12 CHAR) NOT NULL ENABLE,
7     "WYNAGRODZENIE" NUMBER(10,2),
8     | CONSTRAINT "KOORDYNATOR_PK" PRIMARY KEY ("ID_Koordinatator")
9     USING INDEX ENABLE
10    );

```

Tabela 6 Koordynator

LISTA_OCZEKUJACYCH

Columns Data Indexes Constraints Grants Statistics Triggers Dependencies **DDL** Sample Queries

Download Refresh

↻ ↺ 🔍 A↔

```

1 CREATE TABLE "LISTA_OCZEKUJACYCH"
2   ( "ID_OCZEKUJACY" NUMBER(*,0) NOT NULL ENABLE,
3     "DATA_DODANIA" DATE NOT NULL ENABLE,
4     "PODROZE_ID_PODROZY" NUMBER(*,0),
5     "KLIENCI_ID_Klienta" NUMBER(*,0) NOT NULL ENABLE,
6     "ILOSC_OSOB" NUMBER DEFAULT 1,
7     | CONSTRAINT "LISTA_OCZEKUJACYCH_PK" PRIMARY KEY ("ID_OCZEKUJACY")
8     USING INDEX ENABLE
9   );
10
11 ALTER TABLE "LISTA_OCZEKUJACYCH" ADD CONSTRAINT "LISTA_OCZEKUJACYCH_KLIENCI_FK" FOREIGN KEY ("KLIENCI_ID_Klienta")
12   REFERENCES "KLIENCI" ("ID_Klienta") ENABLE;
13 ALTER TABLE "LISTA_OCZEKUJACYCH" ADD CONSTRAINT "LISTA_OCZEKUJACYCH_PODROZE_FK" FOREIGN KEY ("PODROZE_ID_PODROZY")
14   REFERENCES "PODROZE" ("ID_PODROZY") ENABLE;

```

Tabela 7 Lista Oczekujących

PODROZE

Columns Data Indexes Constraints Grants Statistics Triggers Dependencies **DDL** Sample Queries

Download Refresh

↻ ↺ 🔍 A↔

```

7   "DATA_WYJAZDU" DATE NOT NULL ENABLE,
8   "DATA_PRZYJAZDU" DATE NOT NULL ENABLE,
9   "ID_KOORDYNATOR" NUMBER(*,0) NOT NULL ENABLE,
10  "ID_HOTEL" NUMBER(*,0) NOT NULL ENABLE,
11  "POPULARNOSC" NUMBER DEFAULT 0,
12  "PROMOCJA" NUMBER(1,0) DEFAULT 0
13  );
14 CREATE UNIQUE INDEX "UQ_PODROZE_ID" ON "PODROZE" ("ID_PODROZY")
15 ;
16 ALTER TABLE "PODROZE" ADD CONSTRAINT "PODROZE_PK" PRIMARY KEY ("ID_PODROZY")
17   USING INDEX "UQ_PODROZE_ID" ENABLE;
18
19 ALTER TABLE "PODROZE" ADD CONSTRAINT "PODROZE_HOTELE_FK" FOREIGN KEY ("ID_HOTEL")
20   REFERENCES "HOTELE" ("ID_HOTELU") ENABLE;
21 ALTER TABLE "PODROZE" ADD CONSTRAINT "PODROZE_KOORDYNATOR_FK" FOREIGN KEY ("ID_KOORDYNATOR")
22   REFERENCES "KOORDYNATOR" ("ID_KOORDYNATOR") ENABLE;

```

Tabela 8 Podróże

The screenshot shows the Oracle SQL Developer interface. The title bar says "PROGRAM_LOJALNOSCIOWY". Below it is a toolbar with "Columns", "Data", "Indexes", "Constraints", "Grants", "Statistics", "Triggers", "Dependencies", "DDL" (which is selected), and "Sample Queries". There are also "Download" and "Refresh" buttons. Below the toolbar are icons for Undo, Redo, Find, and Paste. The main area contains the following DDL code:

```
1 CREATE TABLE "PROGRAM_LOJALNOSCIOWY"
2   (   "ID_Klienta" NUMBER NOT NULL ENABLE,
3       "ID_Zamowienia" NUMBER NOT NULL ENABLE,
4       "Punkty" NUMBER DEFAULT 0,
5       | PRIMARY KEY ("ID_Klienta", "ID_Zamowienia")
6   USING INDEX ENABLE
7   ) ;
```

Tabela 9 Program Lojalnościowy

Tabela 10 Raporty



ZAMOWIENIA

Columns Data Indexes Constraints Grants Statistics Triggers Dependencies **DDL** Sample Queries

Download Refresh

A::

```
1 CREATE TABLE "ZAMOWIENIA"
2   ( "ID_ZAMOWIENIA" NUMBER(*,0) NOT NULL ENABLE,
3     "DATA_PLATNOSCI" DATE,
4     "KLIENCI_ID_Klienta" NUMBER(*,0),
5     "PODROZE_ID_PODROZY" NUMBER(*,0) NOT NULL ENABLE,
6     "KWOTA" NUMBER DEFAULT 0 NOT NULL ENABLE,
7     "ILOSC_OSOB" NUMBER DEFAULT 1,
8     "STATUS" VARCHAR2(20) DEFAULT 'Zapłacone',
9     | CONSTRAINT "ZAMOWIENIA_PK" PRIMARY KEY ("ID_ZAMOWIENIA")
10    USING INDEX ENABLE
11  );
12
13 ALTER TABLE "ZAMOWIENIA" ADD CONSTRAINT "ZAMOWIENIA_KLIENCI_FK" FOREIGN KEY ("KLIENCI_ID_Klienta")
14   | REFERENCES "KLIENCI" ("ID_Klienta") ENABLE;
15 ALTER TABLE "ZAMOWIENIA" ADD CONSTRAINT "ZAMOWIENIA_PODROZE_FK" FOREIGN KEY ("PODROZE_ID_PODROZY")
16   | REFERENCES "PODROZE" ("ID_PODROZY") ON DELETE CASCADE ENABLE;
```

Tabela 11 Zamówienia

3.2. Uzupełnienie danych w tabelach

ADRES_SZCZEGOLY			
Columns	Data	Indexes	Constraints
+ Insert Row	Columns...	Filter...	Count Rows Load Data Download
	Refresh		
ID_ADRES	KOD_POCZTOWY	MIEJSCOWOSC	ULICA
1	00-001	Warszawa	Ulica A
2	00-002	Kraków	Ulica B
3	00-003	Łódź	Ulica C
4	00-004	Wrocław	Ulica D
5	00-005	Poznań	Ulica E
6	00-006	Gdańsk	Ulica F
7	00-007	Szczecin	Ulica G
8	00-008	Bydgoszcz	Ulica H
9	00-009	Lublin	Ulica I
10	00-010	Katowice	Ulica J
11	FR-75001	Paryż	Rue de Rivoli
12	DE-10115	Berlin	Friedrichstraße
13	IT-00100	Rzym	Via del Corso
14	ES-08001	Barcelona	La Rambla
15	NL-1012	Amsterdam	Damrak
16	BE-1000	Bruksela	Grand-Place

Tabela 13 Adres Szczegóły

ATRAKCJE			
Columns	Data	Indexes	Constraints
+ Insert Row	Columns...	Filter...	Count Rows Load Data Download
	Refresh		
ID_ATRAKCIJ	NAZWA	OPIS	
1	Wycieczka do Zakopanego	Górskie wyprawy po Tatrach	
2	Wizyta w Krakowie	Zwiedzanie Starego Miasta	
3	Wyprawa do Wrocławia	Odwiedzanie Rynku i Hali Stulecia	
4	Rejs po Bałtyku	Rejs statkiem po morzu Bałtyckim	
5	Wizyta w Gdańsku	Zwiedzanie miasta i plażowanie	
6	Piesza wędrówka po Bieszczadach	Górskie wyprawy w pięknych Bieszczadach	
7	Wycieczka do Poznania	Zamek cesarski i Stary Rynek	
8	Kapelińska w Kołobrzegu	Relaks na plaży i kąpiele w morzu	
9	Park Narodowy Biebrzański	Wycieczka przyrodnicza w Biebrzańskim Parku Narodowym	
10	Wizyta w Toruniu	Zamek krzyżacki i Stare Miasto	
11	Odwiedziny w Łodzi	Piotrkowska i Manufaktura	
12	Wizyta w Lublinie	Kultura i historia Lubelszczyzny	
13	Wędrówka po Pieninach	Spływ Dunajcem i piesze szlaki	
14	Górskie wycieczki w Karkonoszach	Wędrówki po Karkonoskim Parkiem Narodowym	
15	Zwiedzanie Wawelu	Zamek Królewski w Krakowie	
16	Wizyta w Białowieskim Parku Narodowym	Odwiedziny w Puszczy Białowieskiej	

Tabela 14 Atrakcje

HOTELE					
Columns	Data	Indexes	Constraints	Grants	Statistics
+ Insert Row	Columns...	Filter...	Count Rows	Load Data	Download
	Refresh				
ID_HOTELU	NAZWA	STANDARD	TELEFON	OPIS	
3	Hotel Majestic	3 gwiazdkowe	9876543212	Hotel w centrum	
4	Hotel Royal	5 gwiazdkowe	9876543213	Luksusowy hotel	
5	Hotel Europa	3-gwiazdkowy	9876543214	Hotel położony w	
6	Hotel Amber	3 gwiazdkowe	9876543215	Hotel nad jeziorem	
7	Hotel Sea View	4 gwiazdkowe	9876543216	Hotel z widokiem na	
8	Hotel Sunrise	3 gwiazdkowe	9876543217	Hotel przy plaży	
9	Hotel Snow	5 gwiazdkowe	9876543218	Ekskluzywny hotel	
10	Hotel Lakeview	4 gwiazdkowe	9876543219	Hotel nad jeziorem	
21	Hotel Mariot	5 Stars	123456789	Luxury hotel w	

Tabela 12 Hotele

KLIENCI								
Columns	Data	Indexes	Constraints	Grants	Statistics	Triggers	Dependencies	DDL
+ Insert Row	Columns...	Filter...	Count Rows	Load Data	Download			
Refresh								
ID_Klienta	NAZWISKO	IMIE	NIP	NAZWA_FIRMY				
4	Jankowski	Marek	4567890123	Firma Jankowski				
6	Zieliński	Adam	6789012345	Firma Zieliński				
7	Szymański	Ewa	7890123456	Firma Szymański				
8	Wójcik	Tomasz	8901234567	Firma Wójcik				
9	Kowalczyk	Karolina	9012345678	Firma Kowalczyk				
10	Kaczmarek	Michał	0123456789	Firma Kaczmarek				
11	Dąbrowski	Krzysztof	1122334455	Firma Dąbrowski				
12	Kwiatkowski	Monika	2233445566	Firma Kwiatko...				
13	Mazur	Jacek	3344556677	Firma Mazur				
14	Woźniak	Joanna	4455667788	Firma Woźniak				
15	Olszewski	Piotr	5566778899	Firma Olszewski				
16	Sikora	Kamil	6677889900	Firma Sikora				
17	Bąk	Marek	7788990011	Firma Bąk				
18	Kaczmarek	Anita	8899001122	Firma Kaczmarek				
19	Baran	Jolanta	9900112233	Firma Baran				
20	Chmiel	Zofia	1011122334	Firma Chmiel				

Tabela 15 Klienci

KOORDYNATOR								
Columns	Data	Indexes	Constraints	Grants	Statistics	Triggers	Dependencies	DDL
+ Insert Row	Columns...	Filter...	Count Rows	Load Data	Download			
Refresh								
ID_KOORDYNATOR	NAZWISKO	IMIE	EMAIL	TELEFON				
3	Wiśniewski	Piotr	piotr.wisniewsk...	9876543212				
4	Kaczmarek	Marek	marek.kaczmar...	9876543213				
5	Wójcik	Ewa	ewa.wojcik@e...	9876543214				
6	Jankowski	Tomasz	tomasz.jankow...	9876543215				
7	Zieliński	Alicja	alicja.zielinski...	9876543216				
8	Szymański	Paweł	pawel.szymans...	9876543217				
9	Mazur	Karolina	karolina.mazur...	9876543218				
10	Dąbrowski	Piotr	piotr.dabrowski...	9876543219				
11	Kwiatkowski	Magdalena	magdalena.kwi...	9876543220				
12	Kamińska	Zofia	zofia.kaminska...	9876543221				
13	Kondrat	Andrzej	andrzej.kondra...	9876543222				
14	Chmiel	Zbigniew	zbigniew.chmi...	9876543223				
15	Adamczyk	Aleksandra	aleksandra.ada...	9876543224				

Tabela 16 Koordynator

PODROZE								
Data	Indexes	Constraints	Grants	Statistics	Triggers	Dependencies	DDL	Sample Queries
+ Insert Row	Columns...	Filter...	Count Rows	Load Data	Download			
Refresh								
ID_PODROZY	KRAJ	ILOSC_DOSTEPNYC	ILOSC_ZAJETYCH_	KOSZT_NA_OSOB				
2	Hiszpania	30	26	200				
3	Włochy	50	35	180				
4	Grecja	30	30	220				
6	Portugalia	45	15	170				
7	Turcja	28	22	210				
8	Egipt	30	20	240				
9	Chorwacja	36	59	160				
10	Szwajcaria	50	10	300				
11	Finlandia	70	30	190				
12	Norwegia	45	10	230				
13	Islandia	30	5	320				
14	Dania	18	7	280				
15	Czechy	15	30	160				

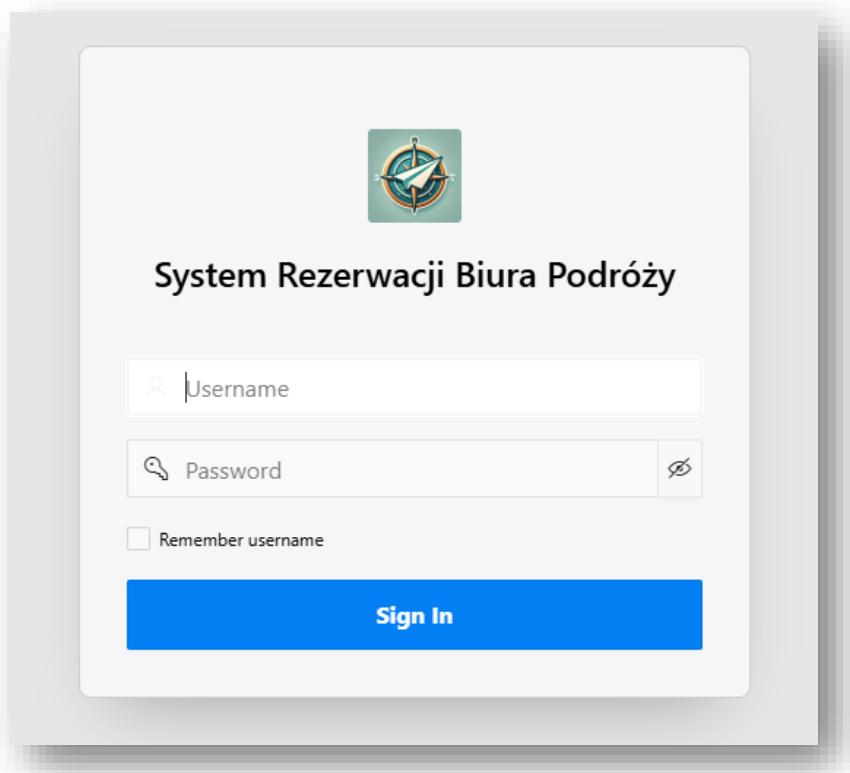
Tabela 18 Podróże

ZAMÓWIENIA								
Data	Indexes	Constraints	Grants	Statistics	Triggers	Dependencies	DDL	Sample Queries
+ Insert Row	Columns...	Filter...	Count Rows	Load Data	Download			
Refresh								
ATNOSCI	KLIENCI_ID_KLIENTA	PODROZE_ID_PODROZY	KWOTA	ILOSC_OSOB	STATUS			
15	143	63	300075	15	Anulowane			
15	141	64	600	3	Zaplacone			
15	12	4	2200	10	Zaplacone			
15	4	3	1200	3	Zaplacone			
15	2	2	6434	1	Zaplacone			
15	4	4	645	1	Anulowane			
25	6	6	867	1	Zaplacone			
25	7	7	234	1	Zaplacone			
25	8	8	435	1	Zaplacone			
25	9	9	567	1	Zaplacone			
25	10	10	200	1	Anulowane			
25	12	2	400	1	Zaplacone			
15	13	3	367	1	Zaplacone			
25	14	4	3565	1	Zaplacone			
5	1	2	768	1	Zaplacone			

Tabela 17 Zamówienia

4. Wygląd Aplikacji

4.1. Strona Logowania



Strona logowania to pierwszy element aplikacji, umożliwiający dostęp do systemu rezerwacji. Posiada intuicyjny układ i proste pola wprowadzania danych:

- **Nazwa użytkownika (Username):** Pole tekstowe do wpisania loginu.
- **Hasło (Password):** Pole tekstowe do wprowadzenia hasła, chronione poprzez ukrywanie wpisywanych znaków.
- **Opcja "Remember username":** Umożliwia zapisanie loginu, co upraszcza kolejne logowania.
- **Przycisk "Sign In":** Rozpoczyna proces logowania i przenosi użytkownika do systemu.

Strona wyróżnia się minimalistycznym designem i dbałością o bezpieczeństwo, wykorzystując szyfrowane połączenie.

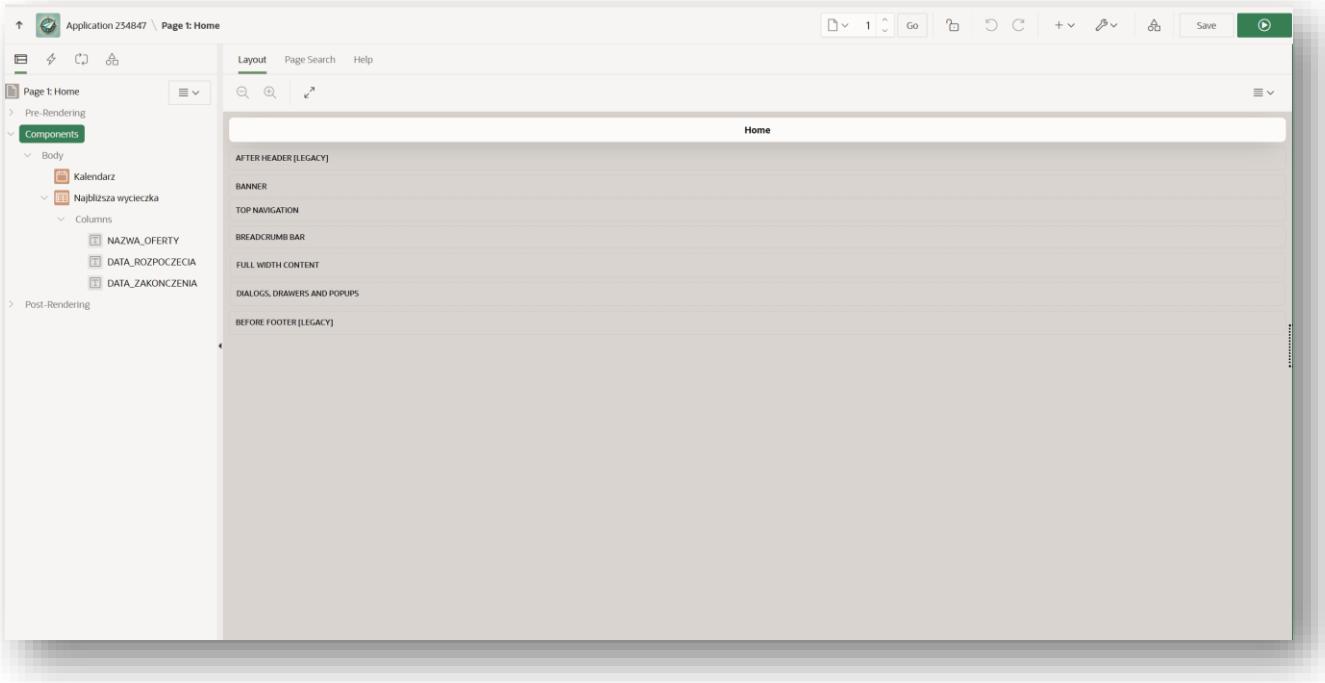
4.2. Strona Główna

The screenshot displays the main dashboard of the application. At the top, there is a horizontal navigation bar with links: Home, Zarządzaj Kordynatorami, Zarządzaj Klientami, Zarządzaj Wycieczkami, Zarządzaj Hotelami, Statystyki, Zarządzaj Rezerwacjami, Platność, Program Lojalnościowy, and Podsumowanie Miesiąca. Below the navigation bar is a monthly calendar for January 2025. The calendar shows days from Monday to Sunday, with specific dates highlighted in blue (26, 27, 28, 29, 30) and yellow (31). A tooltip 'Alja połka' appears over the date 26. Below the calendar, there is a section titled 'Najbliższa Wycieczka' (Next Travel Offer) which lists the offer name 'Hiszpania', start date '2025-03-01', and end date '2025-03-10'. At the bottom of the dashboard, there is a footer bar with icons for search, refresh, user profile, page number 'Page 1', session management, quick edit, customization, and help.

Strona główna pełni funkcję interaktywnego **dashboardu**, który w przejrzysty sposób prezentuje kluczowe informacje dla użytkownika. Znajdują się na niej:

- Kalendarz ofert:** Widok miesięczny umożliwiający przeglądanie szczegółów wycieczek, takich jak nazwa i destynacja. Dostępna jest również opcja "Plan dnia", która pozwala na szczegółowe zaplanowanie wydarzeń.
- Najbliższa wycieczka:** Panel prezentujący daty rozpoczęcia i zakończenia najbliższej wycieczki oraz jej nazwę.
- Menu nawigacyjne:** Szybki dostęp do głównych funkcji aplikacji, takich jak zarządzanie klientami, wycieczkami oraz statystykami.

4.2.1. Struktura strony głównej



1. Kalendarz

- Region typu **Calendar**, wyświetlający dynamicznie dane o ofertach wycieczek z tabel bazy danych.
- Zawiera kolumny:
 - **NAZWA_OFERTY**: Wyświetla nazwę wycieczki.
 - **DATA_ROZPOCZĘCIA**: Pokazuje datę rozpoczęcia wycieczki.
 - **DATA_ZAKOŃCZENIA**: Pokazuje datę zakończenia wycieczki.
- Region umożliwia filtrowanie danych za pomocą wbudowanych opcji nawigacyjnych (Previous/Next).

2. Najbliższa Wycieczka

- Region typu **Interactive Report**, prezentujący listę najbliższych wycieczek.
- Dane pobierane są za pomocą dynamicznych zapytań SQL, co pozwala na automatyczną aktualizację wyników.

4.3. Zarządzanie Koordynatorami

Home	Zarządzaj Koordynatorami	Zarządzaj Klientami	Zarządzaj Wycieczkami	Zarządzaj Hotelami	Statystyki	Zarządzaj Rezerwacjami	Płatność	Program Lojalnościowy	Podsumowanie Miesiąca
Koordynatorzy									
Nazwisko	Imię	Email			Telefon				Wynagrodzenie
Kowalski	Jan	jan.kowalski@email.com			9876543210				5000
Nowak	Anna	anna.nowak@email.com			9876543211				5500
Wiśniewski	Piotr	piotr.wisniewski@email.com			9876543212				6000
Kaczmarek	Marek	marek.kaczmarek@email.com			9876543213				5700
Wojciek	Ewa	ewa.wojciek@email.com			9876543214				5200
Jankowski	Tomasz	tomasz.jankowski@email.com			9876543215				5600
Zielinski	Alicja	alicja.zielinski@email.com			9876543216				5100
Szymański	Paweł	pawel.szymanski@email.com			9876543217				5300
Mazur	Karolina	karolina.mazur@email.com			9876543218				5400
Dąbrowski	Piotr	piotr.dabrowski@email.com			9876543219				6000
Kwiatkowski	Magdalena	magdalena.kwiatkowski@email.com			9876543220				5800
Kamińska	Zofia	zofia.kaminska@email.com			9876543221				5900
Kondrat	Andrzej	andrzej.kondrat@email.com			9876543222				6200
Chmiel	Zbigniew	zbigniew.chmiel@email.com			9876543223				6100
Adamczyk	Aleksandra	aleksandra.adamczyk@email.com			9876543224				6400
z	z	z@z.com			333222111				100
Sobota	Weronika	Werkas123@gmail.com			876678555				8000
w	w	w			123				2000
kwas	waldek	w2w			609253749				2000
adasd	adasdas	w@o2.pl			123321123				2000
Kisiel	Mateusz	mati123@o2.pl			070080090				15000
Strzepek	Katarzyna								7000

Dodaj Koordynatora

Nazwisko	<input type="text"/>
Imię	<input type="text"/>
Email	<input type="text"/>
Telefon	<input type="text"/>
Wynagrodzenie	<input type="text"/>
Dodaj koordynatora	<input type="button"/>

Strona umożliwia pełne zarządzanie danymi koordynatorów, w tym ich dodawanie, przeglądanie oraz edycję.

Funkcjonalności:

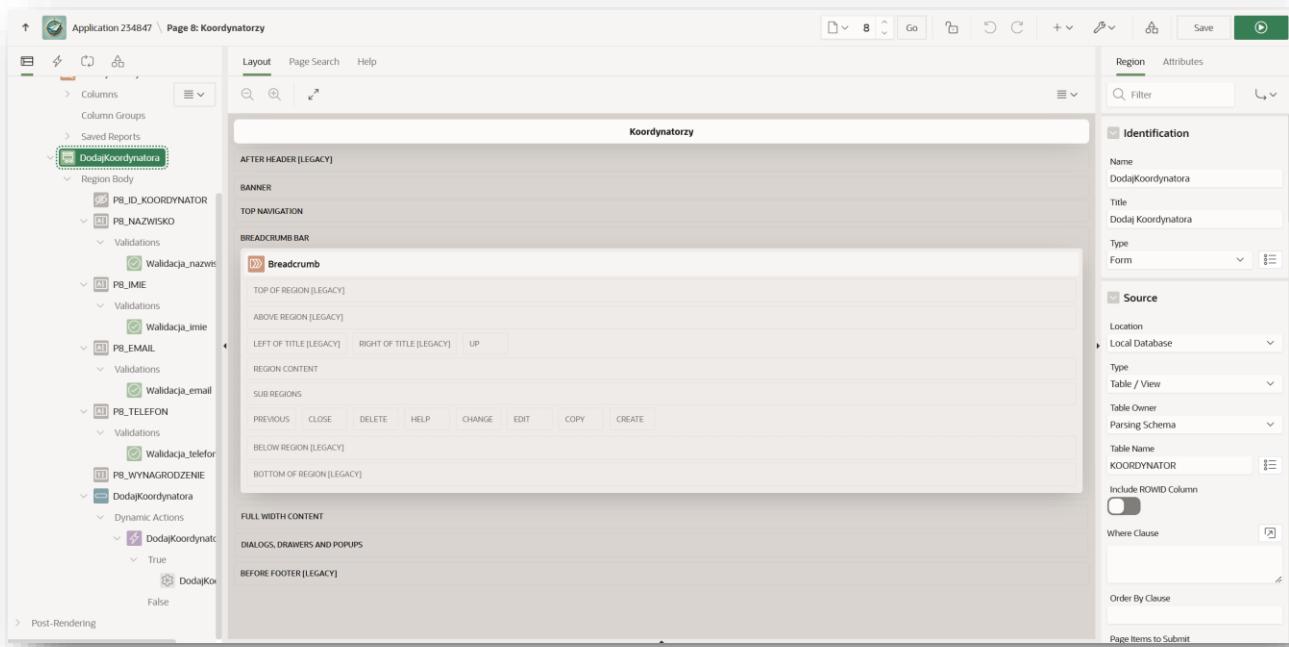
1. Tabela Koordynatorów:

- Wyświetla listę wszystkich koordynatorów wraz z ich danymi, takimi jak nazwisko, imię, e-mail, telefon oraz wynagrodzenie.
- Pozwala na łatwe sortowanie i filtrowanie danych.

2. Formularz Dodawania Koordynatora:

- Użytkownik wprowadza dane koordynatora w pięciu polach: nazwisko, imię, e-mail, telefon i wynagrodzenie.
- Po kliknięciu przycisku "Dodaj koordynatora" dane są wysyłane do bazy przy użyciu dynamicznej akcji w APEX.
-

4.3.1. Struktura strony Koordynatorzy



Strona pozwala na zarządzanie danymi koordynatorów poprzez formularz zintegrowany z tabelą KOORDYNATOR. Pola formularza obsługują dane takie jak nazwisko, imię, email, telefon i wynagrodzenie. Każde pole posiada walidacje (np. wyrażenia regularne dla nazwiska i emaila), co zapewnia poprawność wprowadzanych danych.

Dynamiczna akcja „Dodaj Koordynatora” uruchamia kod PL/SQL dodający dane do tabeli w bazie. Walidacje i automatyczne przetwarzanie danych minimalizują błędy użytkownika, zapewniając prostotę obsługi i spójność danych.

The left panel shows a configuration for a validation rule:

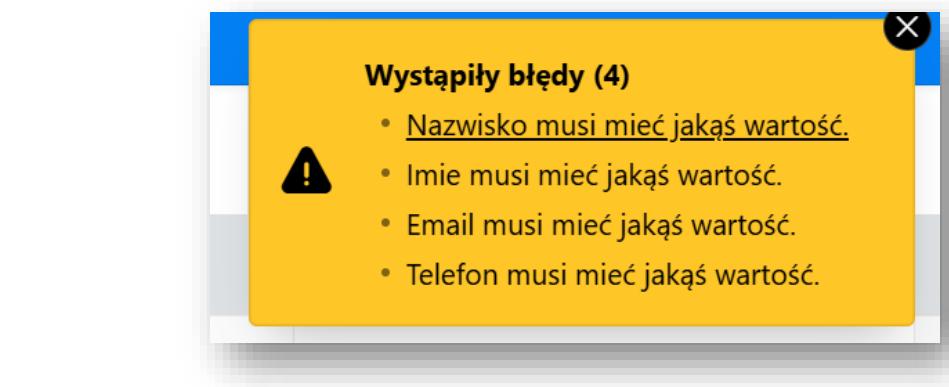
- Editable Region:** - Select -
- Type:** Item matches Regular Express
- Item:** P8_NAZWISKO
- Regular Expression:** ^[A-Za-zA-ZĘŁŃÓÓŚŚŻŻŻŻ]{2,50}\$
- Always Execute:**

The right panel shows a list of validations:

- After Submit
- Validating
 - Validations
 - Walidacja_imie
 - Walidacja_nazwisko** (highlighted with a green border)
 - Walidacja_email
 - Walidacja_telefon
- Processing
- After Processing
- Ajax Callback

Formularz dodawania koordynatora wykorzystuje walidacje, aby zapewnić poprawność danych:

- **Nazwisko (P8_NAZWISKO):** Sprawdzane za pomocą wyrażenia regularnego, dopuszczającego litery, polskie znaki i spacje (2-50 znaków).
- **Pozostale pola (Imię, Email, Telefon):** Walidacje weryfikują format wprowadzonych danych.
- Walidacje uruchamiane są po zatwierdzeniu formularza („After Submit”). W przypadku błędów użytkownik otrzymuje komunikat, a zapis zostaje wstrzymany.



Dodaj Koordynatora

Nazwisko
Michał

Imię
Andrzej

Email
email

Wprowadź poprawny email.

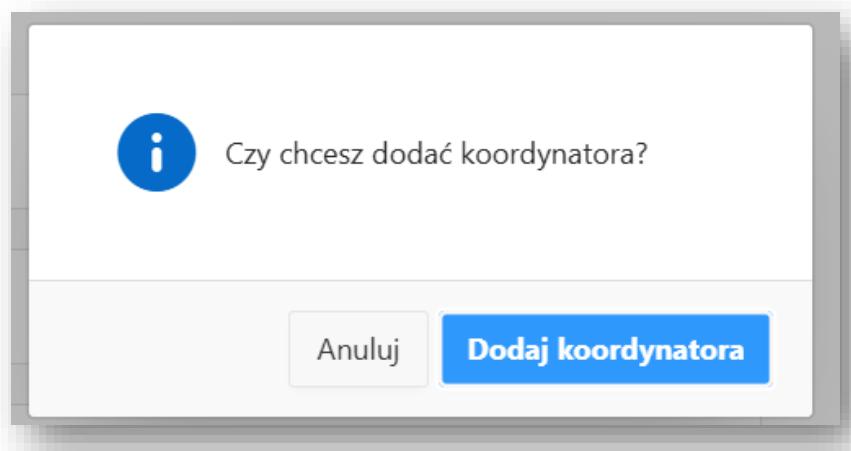
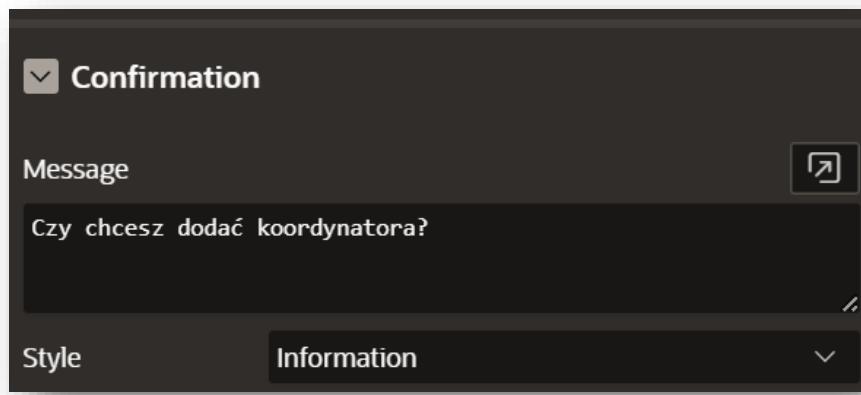
Telefon
601543950

Wynagrodzenie
2000

Dodaj koordynatora

Walidacja jest sprawdzana zarówno na poziomie procedury jak i funkcji walidacji w APEX

Przed każdą operacją CRUD:



4.4. Strona Zarządzanie klientami

Zarządzaj Klientami

Dodaj klienta

Nazwisko

Imię

NIP

Firma

Adres

Kod pocztowy

Miasto

Email

Telefon

Dodaj Klienta

Usuń Klientów

ID klienta

Usuń klineta

Przeglądaj Klientów

Zarządzaj Klientami \

Szukaj: Wszystkie kolumny tekstowe Wykonaj Czynności

Resetuj

Nazwisko	Imię	Nip	Nazwa Firma	Adres	Email	Telefon
Kowalski	Jan	1234567890	Firma Kowalski	1	jan.kowalski@email.com	9876543210
ja	ja	1234	34	70	ja@gmail.com	42525245
jakub	test	test	testowo	35	jakubsobobdwon@gmail.com	124065426
Nowcia	Anna	9876543210	Firma ABC	53	anna.nowak@example.com	987654321
Nowak	Jan	123456789012	Firma Testowa	27	nowak@example.com	848123456
Nowak	Anna	2345678901	Firma Nowak	2	anna.nowak@email.com	9876543211
Jankowski	Marek	4567890123	Firma Jankowski	4	marek.jankowski@email.com	9876543213
Zielinski	Adam	6789012345	Firma Zielinski	6	adam.zielinski@email.com	9876543215
Szymański	Ewa	7890123456	Firma Szymański	7	ewa.szymanski@email.com	9876543216

Opis funkcjonalności:

1. **Dodaj klienta:** Sekcja formularza, w której użytkownik wprowadza dane nowego klienta, takie jak nazwisko, imię, NIP, firma, adres, kod pocztowy, miasto, email oraz telefon. Po uzupełnieniu i kliknięciu przycisku „Dodaj klienta” wykonywana jest dynamiczna akcja „Rejestracja klienta” (PL/SQL), która dodaje dane do bazy.
2. **Usuń klientów:** Sekcja umożliwia usunięcie klienta na podstawie wybranego ID z listy rozwijanej. Po zatwierdzeniu dane klienta są usuwane z bazy.
3. **Przeglądaj klientów:** Tabela wyświetlająca wszystkie aktualne rekordy klientów z możliwością wyszukiwania i filtrowania po kolumnach.

Uproszczona struktura i obsługa dynamicznych akcji gwarantuje łatwe zarządzanie danymi w systemie.

4.4.1. Struktura strony Zarządzanie Klientami

The screenshot displays the structure of an Oracle ADF application titled "Page 5: Zarządzaj Klientami". The interface is divided into two main sections: "Dynamic Actions" on the left and "Processing" on the right.

Dynamic Actions:

- DodajKlienta:** Contains a "New" action with a "True" condition. The condition "Rejestracja klienta" is set to False. It also contains a "New_1" action with a "True" condition, which executes JavaScript code.
- UsunKlientów:** Contains a "Region Body" with a "P5_NEW" component and a "UsuńKlineta" action.
- Przeglądaiklientów:** This section is currently collapsed.

Processing:

- After Submit:** Contains a "Validating" section with a "Validations" subsection. It lists nine validation rules, all of which are checked (indicated by a green checkmark):
 - Walidacja_imie
 - Walidacja_nazwisko
 - Walidacja_telefon
 - Walidacja_email
 - Walidacja_nip
 - Walidacja_firma
 - Walidacja_adres
 - Walidacja_kod_pocztowy
 - Walidacja_miasto
- Processing:** Contains three sub-sections: "Processing", "After Processing", and "Ajax Callback".

1. Dynamic Actions:

- **DodajKlienta:** Dynamiczna akcja wywołująca procedurę Rejestracja klienta (PL/SQL) po wypełnieniu formularza. Zawiera walidacje pól, takich jak imię, nazwisko, telefon, email, i inne wymagane dane.
- **UsunKlientów:** Dynamiczna akcja umożliwiająca usunięcie klienta z bazy danych na podstawie wybranego ID.

2. Validations (Walidacje):

- Walidacje oparte na typach danych i wyrażeniach regularnych zapewniają poprawność wprowadzanych danych, np. format emaila (@), długość pól, oraz zgodność z wymaganiami, np. NIP czy kod pocztowy.

4.5. Strona Zarządzaj wycieczkami

The screenshot displays a web application interface for managing excursions. At the top, there is a navigation bar with various links: Home, Zarządzaj Kordynatorami, Zarządzaj Klientami, Zarządzaj Wycieczkami (highlighted in blue), Zarządzaj Hotelami, Statystyki, Zarządzaj Rezerwacjami, Płatność, Program Lojalnościowy, and Podsumowanie Miesiąca.

The main content area has a title "Zarządzaj wycieczkami". Below it, a sub-section titled "Dodaj wycieczkę" contains several input fields:

- Kraj
- Ilosc Dostepnych Miejsc
- Ilosc Zajetych Miejsc
- Koszt Na Osobe
- Data Wyjazdu
- Data Przyjazdu
- Id Koordynator
- Nazwa atrakcji
- Opis atrakcji

Below this, another section titled "Anuluj wycieczkę" shows a dropdown menu set to "Wybierz wycieczkę" and a yellow "Anuluj wycieczkę" button.

A modal dialog box is open, prompting the user to confirm the cancellation of an excursion. It contains the following text:
Na pewno chcesz usunąć wycieczkę?
(Tego procesu nie możesz cofnąć)

The dialog box has two buttons: "Anuluj" (Cancel) and a red "Anuluj wycieczkę" button.

Wyszukiwanie wycieczek											
Szukaj: Wszystkie kolumny tekstowe		Wykonaj		Czynności							
ID Wycieczki	Kraj Wycieczki	Data Wyjazdu	Data Przyjazdu	Koszt Na Osobę	Dostępne Miejsca	Zajete Miejsca	Nazwa Hotelu	Standard Hotelu	Telefon Hotelu	Nazwa Atrakcji	Opis Atrakcji
2	Hiszpania	2024-07-10	2024-07-20	200	30	26	Hotel Plaza	5 gwiazdkowy	9876543211	Wizyta w Krakowie	Zwiedzanie Starego Miasta
3	Włochy	2024-08-01	2024-08-15	180	50	35	Hotel Plaza	5 gwiazdkowy	9876543211	Wycieczka do Wroclawia	Odwiedzanie Rynku i Hali Targowej
4	Grecja	2024-06-25	2024-07-05	220	30	30	Hotel Royal	5 gwiazdkowy	9876543213	Rejs po Bałtyku	Rejs statkiem po morzu Bałtyckim
6	Portugalia	2024-07-15	2024-07-30	170	45	15	Hotel Plaza	5 gwiazdkowy	9876543211	Piesza wędrówka po ...	Góralska wycieczka w pięciu dniach
7	Turcja	2024-06-28	2024-07-12	210	28	22	Hotel Plaza	5 gwiazdkowy	9876543211	Wycieczka do Poznania	Zamek cesarski i Stary Rynek
8	Egipt	2024-09-01	2024-09-15	240	30	20	Hotel Plaza	5 gwiazdkowy	9876543211	Kapelińska w Kołobrzegu	Relaks na plaży i kąpiel w morzu Bałtyckim
9	Chorwacja	2024-06-10	2024-06-25	160	36	59	Hotel Snow	5 gwiazdkowy	9876543218	Park Narodowy Biebrzański	Wycieczka przyrodnicza po parku narodowym
10	Szwajcaria	2024-08-10	2024-08-20	300	50	10	Hotel Plaza	5 gwiazdkowy	9876543211	Wizyta w Toruniu	Zamek krzyżacki i Stare Miasto
10	Szwajcaria	2024-08-10	2024-08-20	300	50	10	Hotel Plaza	5 gwiazdkowy	9876543211	Wycieczka do Zakopanego	Góralska wycieczka po Tatry
11	Finlandia	2024-07-05	2024-07-20	190	70	30	Hotel Plaza	5 gwiazdkowy	9876543211	Odwiedziny w Łodzi	Piotrkowska i Manufaktura
12	Norwegia	2024-09-05	2024-09-15	230	45	10	Hotel Plaza	5 gwiazdkowy	9876543211	Wycieczka w Lublinie	Kultura i historia Lubelszczyzny
13	Islandia	2024-09-10	2024-09-20	320	30	5	Hotel Plaza	5 gwiazdkowy	9876543211	Wędrówka po Pieninach	Sądeckie Dunajecem i pieszo
14	Dania	2024-07-18	2024-07-28	280	18	7	Hotel Royal	5 gwiazdkowy	9876543213	Góralskie wycieczki w Karkonoszach	Wędrówki po Karkonoszach

Strona „Zarządzaj wycieczkami” umożliwia dodawanie, anulowanie i przeglądanie wycieczek.

1. Kluczowe elementy strony:

2. Dodaj wycieczkę:

- Formularz do wprowadzania szczegółów wycieczki, takich jak kraj, dostępne miejsca, daty, koszt na osobę, ID koordynatora oraz opis atrakcji.
- Po wypełnieniu formularza dane są przesyłane do procedury PL/SQL odpowiedzialnej za rejestrację wycieczki w bazie.

3. Anuluj wycieczkę:

- Wybór wycieczki do anulowania poprzez listę rozwijaną.
- Dynamiczna akcja wywołująca procedurę PL/SQL, która zmienia status wycieczki lub usuwa ją z bazy danych.

4. Przeglądaj wycieczki:

- Interaktywna tabela wyświetlająca listę wszystkich wycieczek z możliwością filtrowania i wyszukiwania.
- Dane obejmują szczegóły, takie jak nazwa wycieczki, daty, koszt, status miejsc oraz przypisany hotel i koordynator.

4.5.1. Struktura strony zarządzaj wycieczkami

The screenshot shows the Oracle ADF Faces Application Designer interface. The top navigation bar includes tabs for 'Rendering', 'Dynamic Actions', 'Processing', and 'Page Shared Components'. The left sidebar contains a tree view of page components:

- Pre-Rendering
- Components
 - Breadcrumb Bar
 - Breadcrumb
 - Body
 - Dodaj wycieczkę
 - Region Body
 - P24_KRAJ
 - P24_ILOSC_DOSTEPNYCH_MIEJSC
 - P24_ILOSC_ZAJETYCH_MIEJSC
 - P24_KOSZT_NA_OSOBE
 - P24_DATA_WYJAZDU
 - P24_DATA_PRZYJAZDU
 - P24_ID_KOORDYNATORA
 - P24_NAZWA_ATRAKCJI
 - P24_OPIS_ATRAKCJI
 - Close
 - CANCEL
 - Change
 - SUBMIT
 - Anuluj wycieczkę
 - Region Body
 - P24_ID_PODROZY
 - CANCEL_TRIP
 - New
 - New

The screenshot shows two views of the Oracle APEX configuration interface. The left view displays the 'Dynamic Actions' section under the 'Events' category, listing various actions like Click, Submit, New, Refresh, and refresh. The right view displays the 'Processing' section, which includes steps for After Submit, Validating, and Processing. The 'Processing' step is expanded, showing options for Run Stored Procedure, Processes, and After Processing. The 'After Processing' step is also expanded, showing options for Branches, Go To Page 1, and Ajax Callback.

1. Dynamic Actions:

- **Dodaj wycieczkę:**
 - Dynamiczna akcja submit wywołuje procedurę PL/SQL do zapisania danych wycieczki w bazie.
 - Walidacje pól formularza są uruchamiane przed przetwarzaniem, aby upewnić się, że dane są kompletne i zgodne z wymaganiami.
- **Anuluj wycieczkę:**
 - Akcja submit uruchamia procedurę CANCEL_TRIP, która zmienia status wycieczki w bazie danych.
 - Po przetworzeniu system odświeża widok i przekierowuje na stronę główną.

2. Processing:

- Walidacja pól formularza, takich jak kraj, daty i koszt, zdefiniowana w sekcji Validating.
- Procedura PL/SQL przetwarza dane wycieczki lub usuwa wybraną wycieczkę z bazy.

3. Rendering:

- Formularz dodawania wycieczki opiera się na predefiniowanych polach z dynamicznym ładowaniem opcji.
- Sekcja przeglądania tabeli z danymi o wycieczkach generowana dynamicznie na podstawie tabeli w bazie danych.

Procesy:

- Po pomyślnym zapisaniu lub anulowaniu wycieczki system odświeża dane i automatycznie przekierowuje użytkownika, co usprawnia interakcję.

4.6. Strona Zarządzaj Hotelami

The screenshot shows a web-based application interface for managing hotels. At the top, there is a navigation bar with several icons and links: Home, Zarządzaj Kordynatorami, Zarządzaj Klientami, Zarządzaj Wycieczkami, Zarządzaj Hotelami (which is the active tab), Statystyki, Zarządzaj Rezerwacjami, Płatność, Program Lojalnościowy, and Podsumowanie Miesiąca. Below the navigation bar, a secondary header reads "Zarządzaj hotelami". The main content area is titled "Dodaj Hotel". It contains seven input fields: "Nazwa" (Name), "Standard (1-5)" (Standard), "Telefon" (Phone), "Opis" (Description), "Kod Pocztowy" (Postal Code), "Miejscowość" (Town/City), and "Ulica" (Street). A green "Dodaj hotel" (Add hotel) button is located at the bottom left of the form.

Strona „Zarządzaj Hotelami” umożliwia dodawanie nowych hoteli poprzez dedykowany formularz.

Formularz pozwala użytkownikowi na wprowadzenie kluczowych danych o hotelu, takich jak nazwa, standard w skali od 1 do 5, numer telefonu, opis, kod pocztowy, miejscowości oraz ulicę. Wszystkie pola są obowiązkowe i posiadają odpowiednie walidacje, które zapewniają poprawność wprowadzanych danych (np. format numeru telefonu czy kodu pocztowego).

Po wypełnieniu formularza i kliknięciu przycisku „Dodaj hotel”, dane są przesyłane do procedury PL/SQL, która obsługuje zapis nowego hotelu w bazie danych. W przypadku wystąpienia błędów walidacji użytkownik otrzymuje odpowiedni komunikat z informacją o konieczności poprawienia danych.

Strona jest intuicyjna w obsłudze i zintegrowana z pozostałymi modułami systemu, co zapewnia spójność zarządzania danymi w aplikacji.

4.6.1. Struktura strony Zarządzaj hotelami

The screenshot shows the Oracle ADF Page Definition Editor interface. The left sidebar lists components under 'Pre-Rendering' and 'Post-Rendering'. The main area displays the page structure with regions like 'AFTER HEADER [LEGACY]', 'TOP NAVIGATION', 'BANNER', 'BREADCRUMB BAR', 'REGION CONTENT', 'SUB REGIONS', 'FULL WIDTH CONTENT', and 'BEFORE FOOTER [LEGACY]'. On the right, the 'Page' tab is selected, showing details for the page: Name (Zarządzaj hotelami), Alias (zarządzaj-hotelami), Title (Zarządzaj hotelami), Page Group (- Select -), Appearance (Page Mode Normal, Page Template Theme Default), and Navigation Menu (Override User Interface Level).

The screenshot shows the 'Dynamic Actions' tab in the Oracle ADF interface. Under the 'Events' section, there are several entries:

- Page Load
- Change (selected)
- Click
 - DodajHotel (True)
 - DodajHotel (False)
- Refresh
 - Refresh (True)
 - Refresh (False)
- Dialog Closed

The screenshot shows the 'Processing' tab in the Oracle ADF interface. Under the 'Validating' section, there is a 'Validations' entry (selected) containing four validation rules:

- Walidacja_nazwa
- Walidacja_standard
- Walidacja_telefon
- Walidacja_opis

Below this, there are sections for 'Processing', 'After Processing', and 'Ajax Callback'.

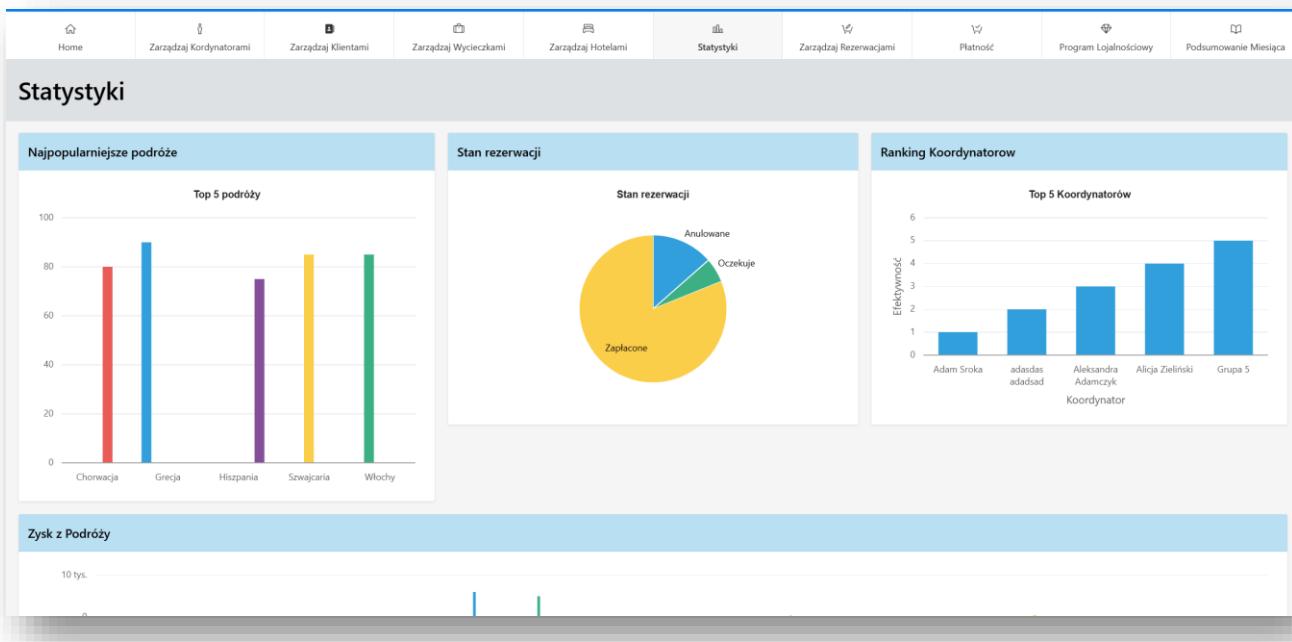
Struktura strony „Zarządzaj hotelami” opiera się na formularzu do wprowadzania danych oraz dynamicznych akcjach i procesach obsługujących zapis w bazie.

Dynamiczna akcja „DodajHotel” wywoływana jest po kliknięciu przycisku „Dodaj hotel”. Proces obejmuje:

- Walidacje pól: Walidacja_nazwa, Walidacja_standard, Walidacja_telefon oraz Walidacja_opis, które zapewniają poprawność wprowadzanych danych. Przykładowo, pole „Telefon” wymaga numeru w odpowiednim formacie.
- Przesyłanie danych do procedury PL/SQL, która dokonuje zapisu nowego hotelu do bazy danych.
- Odświeżenie widoku w celu uwidocznienia wprowadzonych zmian.

Cała logika strony została zaprojektowana tak, aby minimalizować błędy użytkownika i zapewnić intuicyjną obsługę.

4.7. Strona Statystyki



Strona „Statystyki” przedstawia dynamiczne wizualizacje danych, umożliwiające użytkownikowi analizę kluczowych informacji w systemie.

Elementy strony:

- Najpopularniejsze podróże:** Wykres słupkowy prezentuje pięć najczęściej wybieranych kierunków podróży.
- Stan rezerwacji:** Wykres kołowy ilustruje podział rezerwacji na zapłacone, oczekujące oraz anulowane.
- Ranking koordynatorów:** Wykres słupkowy przedstawia pięciu najbardziej efektywnych koordynatorów.
- Zysk z podróży:** Sekcja przedstawia przychody w zależności od zrealizowanych wycieczek.

Wszystkie dane są dynamicznie pobierane z bazy Oracle, a wykresy generowane w czasie rzeczywistym za pomocą komponentów raportowych w Oracle APEX. Układ strony został zoptymalizowany pod kątem czytelności i intuicyjności.

4.7.1. Struktura strony

The screenshot shows the Oracle ADF Page Editor interface. On the left, there's a tree view of page components under 'Page & Statystyki'. Components listed include Pre-Rendering, Components (Breadcrumb Bar, Breadcrumb), Body (Najpopularniejsze podróże, Stan rezerwacji, Ranking Koordynatorów, Zysk z Podróży), and Top Navigation. The main workspace displays the 'Statystyki' page structure with regions like AFTER HEADER [LEGACY], TOP NAVIGATION, BREADCRUMB BAR, BODY, and BOTTOM OF REGION [LEGACY]. The right panel contains sections for Identification (Name: Statystyki, Alias: statystyki, Title: Statystyki, Page Group: - Select -), Appearance (Page Mode: Normal, Page Template: Theme Default), and Navigation Menu (Override User Interface Level). Buttons at the top include Save, Print, and Refresh.

This screenshot shows the Oracle ADF Page Editor for 'Page 6: Statystyki'. The left sidebar lists regions: After Submit, Validating, Processing (Processes: New, New_1), After Processing, and Ajax Callback. The main workspace shows the page structure with regions like AFTER HEADER [LEGACY], TOP NAVIGATION, BREADCRUMB BAR, BODY, and BOTTOM OF REGION [LEGACY]. The right panel includes sections for Identification, Appearance (Page Mode: Normal, Page Template: Theme Default), and Navigation Menu. Buttons at the top include Save, Print, and Refresh.

1. Regiony wykresów:

- **Najpopularniejsze podróże:** Dane są przedstawiane za pomocą wykresu słupkowego, skonfigurowanego na podstawie dynamicznie generowanego zapytania SQL.
- **Stan rezerwacji:** Wykres kołowy, który odzwierciedla proporcje zapłaconych, oczekujących i anulowanych rezerwacji.
- **Ranking koordynatorów:** Słupkowy wykres z ocenami efektywności poszczególnych koordynatorów.
- **Zysk z podróży:** Linia przychodów prezentowana na wykresie.

2. Procesy:

- **Processing:** Dwa procesy „New” i „New_1” obsługujące przetwarzanie danych dla regionów wykresów. Procesy bazują na procedurach PL/SQL, które generują zestawienia na potrzeby wizualizacji.

4.8. Strona Zarządzaj Rezerwacjami

Id Zamówienia	Data Platności ↑	Kwota	Ilość Osób	Status	Klient	Wycieczka	Hotel	Standard Hotelu
5	2025-01-20	1200	10	Zapłacone	Marek Jankowski	Włochy (2024-08-01)	Hotel Plaza	5 gwiazdkowy
132	2025-01-22	420	2	Zapłacone	Katarzyna Strzepek	Turcja (2024-06-28)	Hotel Plaza	5 gwiazdkowy
22	2025-01-22	256	1	Anulowane	Jan Kowalski	Grecja (2024-06-25)	Hotel Royal	5 gwiazdkowy
134	2025-01-22	10002,5	5	Zapłacone	jedno jest	Polska (2025-09-06)	Hotel Green	4 gwiazdki
152	2025-01-22	1600	10	Zapłacone	jedno jest	Chorwacja (2024-06-10)	Hotel Snow	5 gwiazdkowy
151	2025-01-22	320	2	Zapłacone	Jan Kowalski	Chorwacja (2024-06-10)	Hotel Snow	5 gwiazdkowy

Strona „Zarządzaj Rezerwacjami” umożliwia obsługę rezerwacji wycieczek.

- Dodaj Rezerwację:** Formularz pozwala na wybór podróży, klienta oraz określenie liczby osób uczestniczących w rezerwacji. Po wypełnieniu formularza i kliknięciu „Dodaj rezerwację” dane są przesyłane do bazy w celu zapisania nowej rezerwacji.
- Przegląd Rezerwacji:** Tabela wyświetla szczegóły dotyczące istniejących rezerwacji.
- Anuluj Rezerwację:** Pole wyboru umożliwia anulowanie rezerwacji, co zmienia jej status w bazie danych.

4.8.1. Struktura zarządzaj rezerwacjami

The screenshot shows the Oracle APEX App Builder interface. On the left, the navigation pane displays the page structure:

- Pre-Rendering
- Components
 - Breadcrumb Bar
 - Breadcrumb
 - Edit
 - AnulujRezerwacje
 - Body
 - Dodaj Rezerwację
 - Region Body
 - P7 PODROZ
 - P7_KLIENTI
 - P7_LICZBA
 - DODAJREZERWACJE
 - New
 - Columns
 - Column Groups
 - Saved Reports
 - Post-Rendering

The main workspace shows the page structure with a breadcrumb component selected. The breadcrumb configuration includes regions like TOP REGION [LEGACY], ABOVE REGION [LEGACY], LEFT OF TITLE [LEGACY], RIGHT OF TITLE [LEGACY], UP, and DOWN. Buttons for PREVIOUS, CLOSE, DELETE, HELP, CHANGE, EDIT, COPY, and CREATE are visible. The right sidebar contains the Identification and Appearance sections.

The screenshot shows the Oracle APEX Page Properties dialog, specifically the Events section. The tree view shows the following events:

- Page Load
- Change
- Click
 - New
 - True
 - Execute Server-side Code
 - False
 - refresh
 - True
 - refresh
 - False
- Dialog Closed

1. Dodaj Rezerwację:

- Formularz umożliwia wybór podróży, klienta oraz wprowadzenie liczby osób.
- Po wypełnieniu formularza i zatwierdzeniu, rezerwacja jest zapisywana w bazie danych.

2. Anulowanie Rezerwacji:

- Funkcja dostępna w dedykowanym formularzu, gdzie można wybrać rezerwację do anulowania.

3. Lista Rezerwacji:

- Interfejs tabelaryczny umożliwia przeglądanie istniejących rezerwacji, w tym szczegóły dotyczące statusu, daty płatności, kwoty i standardu hotelu.

4.9. Strona Płatności

The screenshot shows a payment interface. At the top, there's a navigation bar with links: Home, Zarządzaj Kordynatorami, Zarządzaj Klientami, Zarządzaj Wycieczkami, Zarządzaj Hotelami, Statystyki, Zarządzaj Rezerwacjami, Płatność (selected), Program Lojalnościowy, and Podsumowanie Miesiąca. Below the navigation is a section titled "Płatność". It contains a dropdown menu showing "Zamówienie 113 | Klient: jedno jest | Kwota: 2200 zł". There are input fields for "Koszt" (2200), "Data" (2025-01-27), and "Punkty" (580). A checkbox "Użyj punktów" is checked. Below these is a "New Kwota" input field and a green "Zapłać" button.

The screenshot shows a confirmation page with a green "Zapłać" button at the top. Below it is a table with a header "New" and columns: Id Zamówienia, Klient, Podroz, Należna Kwota, and Status. The table contains two rows:

Id Zamówienia ↑	Klient	Podroz	Należna Kwota	Status
113	jedno jest	Grecja	2200	Oczekuje
131	Katarzyna Strzępek	Turcja	420	Oczekuje

At the bottom right of the table area, there is a small text "1 - 2".

1. Formularz Płatności:

- Pozwala użytkownikowi wybrać zamówienie z listy, wprowadzić kwotę płatności oraz termin.
- Opcjonalnie można dodać notatki dotyczące płatności.

2. Lista Zamówień:

- Wyświetla zamówienia z informacjami o kliencie, podróży, należnej kwocie i statusie płatności (np. "Oczekuje").

3. Przycisk „Zapłać”:

- Aktywuje proces zatwierdzania płatności, który aktualizuje status zamówienia w bazie danych.

4.9.1. Struktura strony płatności

The screenshot shows the Oracle ADF Page Definition Editor interface. The left sidebar lists page components: Pre-Rendering, Components (Breadcrumb Bar, Oplaty), Body (Platność, Region Body with sub-components like ID_REZERWACJI, ID_Klienta, ZAMOWIENIE, KOSZT, DATA, PUNKTY, UZYJPUNKTY, NEW_KWOTA, Zapłac), New, Columns, and Post-Rendering. The main workspace displays the page structure with regions: AFTER HEADER [LEGACY], BANNER, TOP NAVIGATION, BREADCRUMB BAR, Oplaty (TOP OF REGION [LEGACY]), ABOVE REGION [LEGACY], LEFT OF TITLE [LEGACY], RIGHT OF TITLE [LEGACY], UP, REGION CONTENT, SUB REGIONS (PREVIOUS, CLOSE, DELETE, HELP, CHANGE, EDIT, COPY, CREATE), BELOW REGION [LEGACY], and BOTTOM OF REGION [LEGACY]. The right sidebar contains sections for Identification (Name: Platność, Alias: płatność, Title: Platność, Page Group: - Select -), Appearance (Page Mode: Normal, Page Template: Theme Default, Template Options: Use Template Defaults, CSS Classes, Media Type), and Navigation Menu (Override User Interface Level).

The screenshot shows the Oracle ADF Page Load Actions dialog. It lists actions fired on page load: Page Load (New_1, True, Set Value, False), Change (Set KWOTA Value, New_2, PUNKTY), Click (New), and Dialog Closed.

- > Actions Fired on Page Load
- < Events
 - < Page Load
 - < New_1
 - < True
 - Set Value
 - False
 - < Change
 - < Set KWOTA Value
 - < New_2
 - < PUNKTY
 - < Click
 - < New
 - < Dialog Closed

Dynamiczne Akcje:

1. Page Load:

- Akcja inicjalizuje stronę poprzez ustawienie domyślnych wartości pól, np. przeliczenie kwoty lub ustawienie punktów klienta.

2. Change:

- Wywoływane, gdy użytkownik zmienia dane w formularzu, np. wartość kwoty lub wybrane zamówienie:
 - Akcja aktualizuje pola zależne, takie jak punkty lojalnościowe lub status zamówienia.

3. Kliknięcie Przycisku „Zapłać”:

- Wywołuje akcję serwerową:
 - Waliduje dane płatności.
 - Aktualizuje status zamówienia w bazie na „Zapłacone”.
 - Odświeża tabelę z zamówieniami, aby pokazać najnowsze dane.

4.10. Strona Program Lojalnościowy

The screenshot shows a web application interface for managing a loyalty program. At the top, there is a navigation bar with icons and links: Home, Zarządzaj Kordynatorami, Zarządzaj Klientami, Zarządzaj Wycieczkami, Zarządzaj Hotelami, Statystyki, Zarządzaj Rezerwacjami, Płatność, Program Lojalnościowy (selected), and Podsumowanie Miesiąca.

The main content area is titled "Program Lojalnościowy". It contains three sections:

- Lista Zamówień**: A table showing order details. Columns include Id Zamówienia, Klient, Kwota, and Status. Data rows are as follows:

ID Zamówienia	Klient	Kwota	Status
1	Jan Kowalski	2000	Zapłacone
2	Anna Nowak	6434	Zapłacone
4	Marek Jankowski	645	Anulowane
5	Marek Jankowski	1200	Zapłacone
6	Adam Zieliński	867	Zapłacone
7	Ewa Szymańska	234	Zapłacone
8	Tomasz Wójcik	435	Zapłacone
9	Karolina Kowalczyk	567	Zapłacone
10	Michał Kaczmarek	200	Anulowane
11	Marek Jankowski	1200	Zapłacone
12	Monika Kwiatkowska	400	Zapłacone
13	Jacek Mazur	367	Zapłacone
14	Joanna Woźniak	3565	Zapłacone
15	Jan Kowalski	3245	Zapłacone
20	Jan Kowalski	768	Zapłacone
- Przyznaj Punkty**: A section for assigning points. It includes a dropdown menu for selecting an order and a button labeled "Przyznaj Punkty".
- Lista Punktów Lojalnościowych**: A table showing the total points for each client. Columns include Id Klienta, Klient, and Suma Punktów. Data rows are as follows:

ID Klienta	Klient	Suma Punktów
140	jedno jest	580
141	Katarzyna Strzepek	351
2	Anna Nowak	321
12	Monika Kwiatkowska	290
4	Marek Jankowski	151
1	Jan Kowalski	144
14	Joanna Woźniak	52
6	Adam Zieliński	43
9	Karolina Kowalczyk	28
13	Jacek Mazur	18
21	Jan Nowak	0
24	Jan Nowak	0
82	test jakub	0
83	test jakub	0
84	test jakub	0

Strona przedstawia szczegóły związane z programem lojalnościowym dla klientów. Jest podzielona na trzy główne sekcje:

- Lista Zamówień**: Wyświetla dane zamówień klientów. Informacje te pomagają w weryfikacji, czy dane zamówienie kwalifikuje się do naliczenia punktów lojalnościowych.
- Przyznaj Punkty** (sekcja redundantna): Pozwala wybrać zamówienie z listy w celu manualnego przypisania punktów lojalnościowych. **W praktyce punkty są automatycznie przyznawane procedurą po dokonaniu płatności**, co czyni tę sekcję zbędną.
- Lista Punktów Lojalnościowych**: Prezentuje sumę punktów lojalnościowych zgromadzonych przez poszczególnych klientów.

4.10.1. Struktura strony Program Lojalnościowy

The screenshot shows the Oracle ADF Page Definition Editor interface. The left sidebar displays the page structure:

- Page 11: Płatność
 - Pre-Rendering
 - Components
 - Breadcrumb Bar
 - Oplaty
 - Body
 - Płatność
 - Region Body
 - ID_REZERWACJI
 - ID_Klienta
 - ZAMOWIENIE
 - KOSZT
 - DATA
 - PUNKTY
 - UZYJPUNKTY
 - NEW_kwota
 - Zapłać
 - New
 - Columns
 - Post-Rendering

The main panel shows the page structure with regions and components:

 - AFTER HEADER [LEGACY]
 - BANNER
 - TOP NAVIGATION
 - BREADCRUMB BAR
 - Oplaty
 - REGION BODY
 - TOP OF REGION [LEGACY]
 - ABOVE REGION [LEGACY]
 - LEFT OF TITLE [LEGACY] RIGHT OF TITLE [LEGACY] UP
 - REGION CONTENT
 - SUB REGIONS
 - PREVIOUS CLOSE DELETE HELP CHANGE EDIT COPY CREATE
 - BELOW REGION [LEGACY]
 - BOTTOM OF REGION [LEGACY]
 - FULL WIDTH CONTENT
 - DIALOGS, DRAWERS AND POPUPS
 - BEFORE FOOTER [LEGACY]

The screenshot shows the configuration for actions fired on page load:

- Actions Fired on Page Load
- Events
 - Page Load
 - New_1
 - True
 - Set Value
 - False
 - Change
 - Set KWOTA Value
 - New_2
 - PUNKTY
 - Click
 - New
- Dialog Closed

1. **Formularz Płatności:** Sekcja pozwala na wprowadzenie szczegółów płatności, takich jak:
 - Kwota należna.
 - Data płatności.
 - Informacje o karach finansowych (jeśli dotyczy).
2. **Tabela Zamówień:** Wyświetla listę zamówień oczekujących na płatność, zawierając takie dane jak ID zamówienia, klient, kwota należna oraz status.
3. **Dynamiczne Akcje**
 - **Page Load (Ładowanie strony):**
 - Ustawia wartości domyślne dla pól formularza za pomocą akcji *Set Value*.
 - **Change (Zmiana wartości):**
 - Automatycznie aktualizuje kwoty lub inne parametry w oparciu o zmiany dokonane przez użytkownika.
 - **Click (Kliknięcie):**
 - Inicjuje proces płatności poprzez akcję *Execute Server-side Code*, zapisując szczegóły w bazie danych.

4.11. Strona Podsumowanie Miesiąca

Data	Typ Raportu	Zawartość Raportu
2025-01-22	Podsumowanie miesiąca	Podsumowanie dla miesiąca: 1/2025 Przychód: 30547,5 PLN Liczba zamówień: 12 Łączna liczba osób: 72
2025-01-22	Podsumowanie miesiąca	Podsumowanie dla miesiąca: 1/2025 Przychód: 30547,5 PLN Liczba zamówień: 12 Łączna liczba osób: 72
2025-01-22	Podsumowanie miesiąca	Podsumowanie dla miesiąca: 1/2025 Przychód: 30547,5 PLN Liczba zamówień: 12 Łączna liczba osób: 72
2025-01-22	Podsumowanie miesiąca	Podsumowanie dla miesiąca: 6/2025 Przychód: 0 PLN Liczba zamówień: 0 Łączna liczba osób: 0
2025-01-21	Podsumowanie miesiąca	Podsumowanie dla miesiąca: 4/2025 Przychód: 0 PLN Liczba zamówień: 0 Łączna liczba osób: 0
2025-01-21	Podsumowanie miesiąca	Podsumowanie dla miesiąca: 3/2025 Przychód: 12869 PLN Liczba zamówień: 8 Łączna liczba osób: 8
2025-01-21	Podsumowanie miesiąca	Podsumowanie dla miesiąca: 1/2025 Przychód: 1200 PLN Liczba zamówień: 1 Łączna liczba osób: 10
2025-01-21	Podsumowanie miesiąca	Podsumowanie dla miesiąca: 1/2025 Przychód: 1200 PLN Liczba zamówień: 1 Łączna liczba osób: 10
2025-01-21	Podsumowanie miesiąca	Podsumowanie dla miesiąca: 2/2025 Przychód: 0 PLN Liczba zamówień: 0 Łączna liczba osób: 0
2025-01-21	Podsumowanie miesiąca	Podsumowanie dla miesiąca: 2/2025 Przychód: 0 PLN Liczba zamówień: 0 Łączna liczba osób: 0
2025-01-21	Podsumowanie miesiąca	Podsumowanie dla miesiąca: 2/2025 Przychód: 0 PLN Liczba zamówień: 0 Łączna liczba osób: 0
2025-01-21	Podsumowanie miesiąca	Podsumowanie dla miesiąca: 3/2025 Przychód: 12869 PLN Liczba zamówień: 8 Łączna liczba osób: 8

Strona umożliwia generowanie i przegląd raportów miesięcznych, zawierających kluczowe informacje dotyczące przychodów, liczby zamówień oraz liczby klientów.

- **Tabela Raportów:** Wyświetla listę istniejących raportów z informacją o dacie raportu, jego nazwie oraz szczegółowym opisem.
- **Opcje Akcji:** Umożliwiają wyszukiwanie, filtrowanie oraz eksportowanie danych z tabeli.
- **Procesy**
- **Generowanie Raportów:**
 - Procedura PL/SQL tworzy podsumowanie na podstawie danych finansowych i rezerwacji z bazy danych.
 - Wynik zapisywany jest w tabeli systemowej raportów.

Dynamiczne Akcje

- **Page Load:**
 - Automatyczne ładowanie istniejących raportów i ich wyświetlenie w tabeli.
- **Click:**
 - Akcja generuje nowy raport i odświeża tabelę.

4.11.1. Struktura strony Podsumowanie miesiąca

The screenshot shows the Oracle ADF Page Definition Editor interface. The left pane displays the page structure tree, which includes sections like Pre-Rendering, Components, Body, and Post-Rendering. The Body section contains a 'Podsumowanie Miesiąca' component, which is expanded to show its sub-components: Columns, Saved Reports, and a 'Raport Popularności Wykazów' component. The 'Raport Popularności Wykazów' component has its own sub-components: Columns and 'GenerujRaport'. The right pane shows the 'Page' properties panel, where the page's name is set to 'Podsumowanie Miesiąca' and its alias to 'podsumowanie-miesiąca1'. Other properties like Page Mode (Normal), Page Template (Theme Default), and Navigation Menu are also visible.

The screenshot shows the Oracle ADF Event Definition Editor interface. It displays two panels: one for 'Events' and one for 'Processes'. In the 'Events' panel, under the 'Change' section, there is a 'Click' event associated with a 'generujRaport' button. This event has a single handler: 'True', which triggers a 'Refresh' action. In the 'Processes' panel, there are two processes: 'Generuj_Podsumowanie' and 'GenerujRaport'. These processes are triggered by the 'After Processing' event, which is associated with the 'generujRaport' button's click event.

1. Tabela z Raportami:

- Źródło danych: Tabela z bazy danych, zawierająca informacje o raportach miesięcznych.
- Kolumny: Data raportu, nazwa, zawartość raportu.
- Akcje: Możliwość wyszukiwania i sortowania danych.

2. Dynamiczne Akcje:

• Page Load:

- Automatyczne załadowanie listy raportów na stronie.

• Click – generujRaport:

- Uruchamia proces tworzenia nowego raportu.
- Odświeża tabelę, aby natychmiast pokazać nowo dodany raport.

Procesy

1. Generuj_Podsumowanie:

- Typ: Procedura PL/SQL.
- Funkcjonalność: Generuje nowe podsumowanie na podstawie bieżących danych w systemie.
- Zapisuje dane w tabeli raportów.

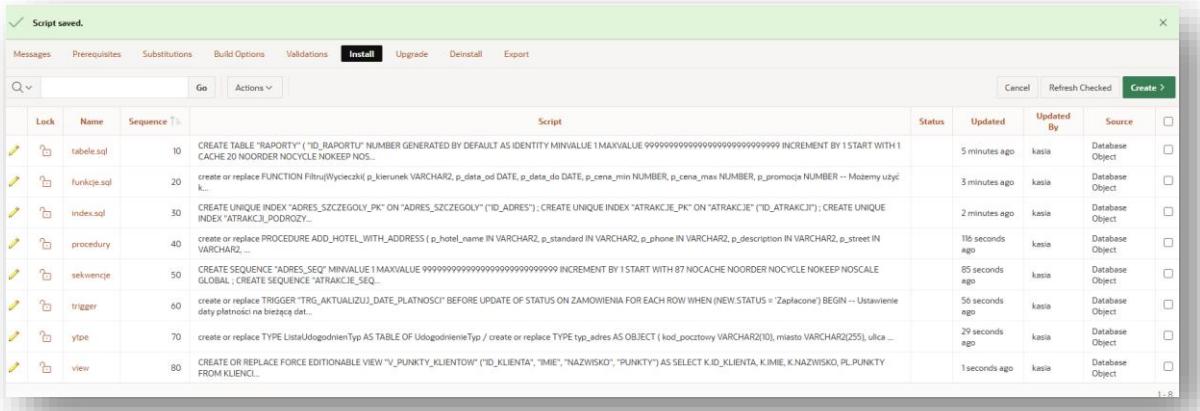
2. GenerujRaport:

- Typ: Proces przetwarzający.
- Funkcjonalność: Odpowiada za wywołanie logiki PL/SQL, która tworzy i zapisuje dane raportu.

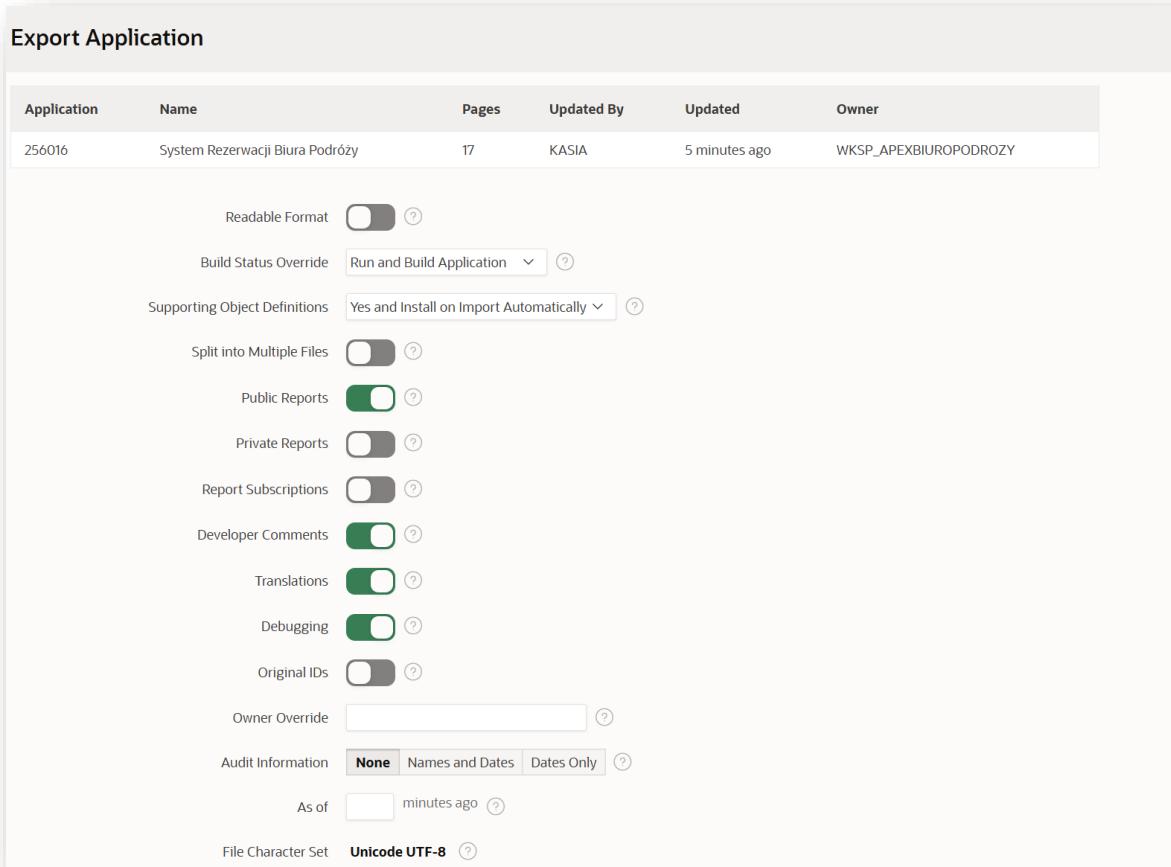
5. Export i Import

5.1. Export

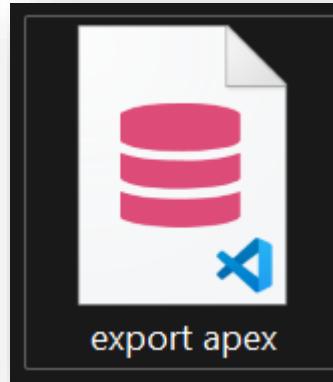
Zaczynamy od utworzenia Installation Scripts (App Builder/aplikacja/Supporting Objects/Installation Scripts) czyli skryptów do importu obiektów i zawartości tabel.



Kolejnym krokiem jest wejście w okno Export i wybranie aplikacji, ważne jest aby zaznaczyć opcje „Yes and Install on Import Automatically”.

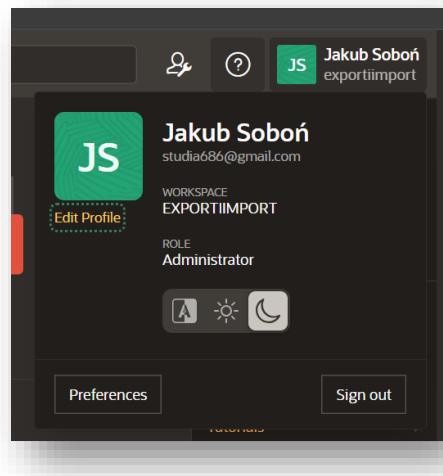


```
f234847.sql >
C: > Users > jakub > Downloads > f234847.sql
66 -- Instance ID: 63113759365424
67 --
68
69 prompt --application/delete_application
70 begin
71 wwv_flow_imp.remove_flow(wwv_flow.g_flow_id);
72 end;
73 /
74 prompt --application/create_application
75 begin
76 wwv_flow_workspace.create_flow(
77 | p_ids>wwv_flow.g_flow_id
78 ,p_owner>nvl(wwv_flow_application_install.get_schema,'WKSP_APEXBIUROPODROZY')
79 ,p_name>nvl(wwv_flow_application_install.get_application_name,unistr('System Rezerwacji Biura Podr\00F3\017Cy'))
80 ,p_alias>nvl(wwv_flow_application_install.get_application_alias,unistr('SYSTEM-REZERWACJI-BIURA-PODR\00D3\017BY'))
81 ,p_page_view_logging>'YES'
82 ,p_page_protection_enabled_y_n>'Y'
83 ,p_checksum_salt>'459F30507839690193E58AD85133F61BEE988244BF412A05D8703C1F8AF15CF1'
84 ,p_bookmark_checksum_function>'SH512'
85 ,p_compatibility_mode>'21.2'
86 ,p_flow_language>'pl'
87 ,p_flow_language_derived_from>'FLOW_PRIMARY_LANGUAGE'
88 ,p_allow_feedback_y_n>'Y'
89 ,p_date_format>'DS'
90 ,p_timestamp_format>'DS'
91 ,p_timestamp_tz_format>'N'
92 ,p_direction_right_to_left>'N'
93 ,p_flow_image_prefix => nvl(wwv_flow_application_install.get_image_prefix,'')
94 ,p_authentication_id>wwv_flow_imp.id(105074801043499286178)
95 ,p_application_tab_set>1
96 ,p_logo_type>'I'
97 ,p_logo>'#APP_FILES#app-234847-logo.jpg'
98 ,p_public_user>'APEX_PUBLIC_USER'
99 ,p_proxy_server>nvl(wwv_flow_application_install.get_proxy,'')
100 ,p_no_proxy_domains>nvl(wwv_flow_application_install.get_no_proxy_domains,'')
101 ,p_flow_version>'Release 1.0'
102 ,p_flow_status>'AVAILABLE_W_EDIT_LINK'
103 ,p_flow_unavailable_text>"This application is currently unavailable at this time."
104 ,p_exact_substitutions_only>'Y'
```

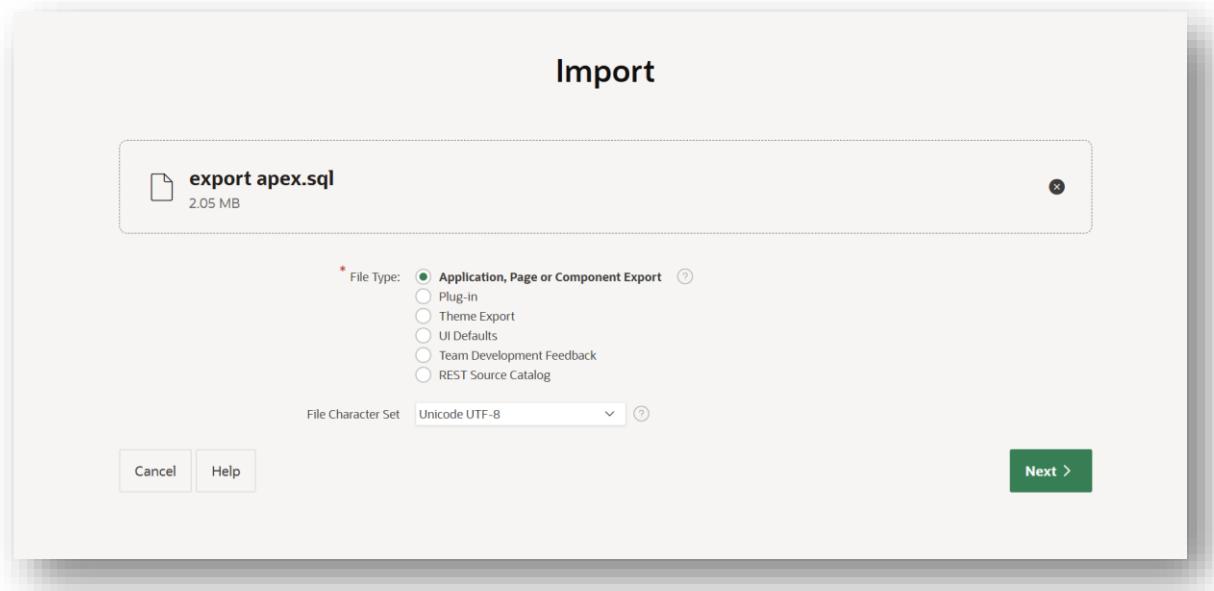


5.2. Import

Zaczynamy od utworzenia nowego konta i workspace'a



W App Builder opcja Import wklejamy nasz wyeksportowany plik i przechodzimy przez proces importu



APEX App Builder SQL Workshop Team Development Gallery

Imported file uploaded.

Install Application

Install Application

Review the installation details and select **Install Application** to install the imported application to your workspace.

Imported Application ID: **234847**

Imported Application Name: **System Rezerwacji Biura Podróży**

Imported Application Version: **2024.11.30**

Imported Application Parsing Schema: **WKSP_APEXBIUROPODROZY**

Imported Application Origin: **This application was exported from the current workspace.**

Imported Application Workspace: **APEXBIUROPODROZY**

Imported Application Workspace ID: **7847749634296508057**

Current Workspace: **APEXBIUROPODROZY**

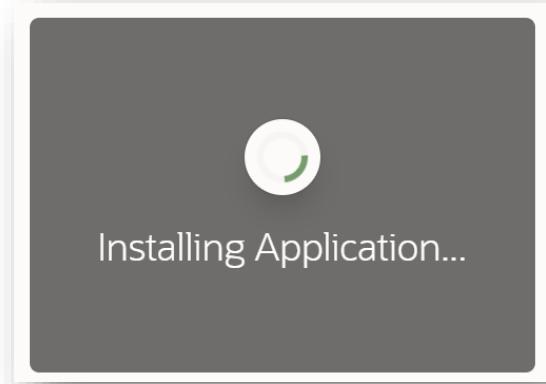
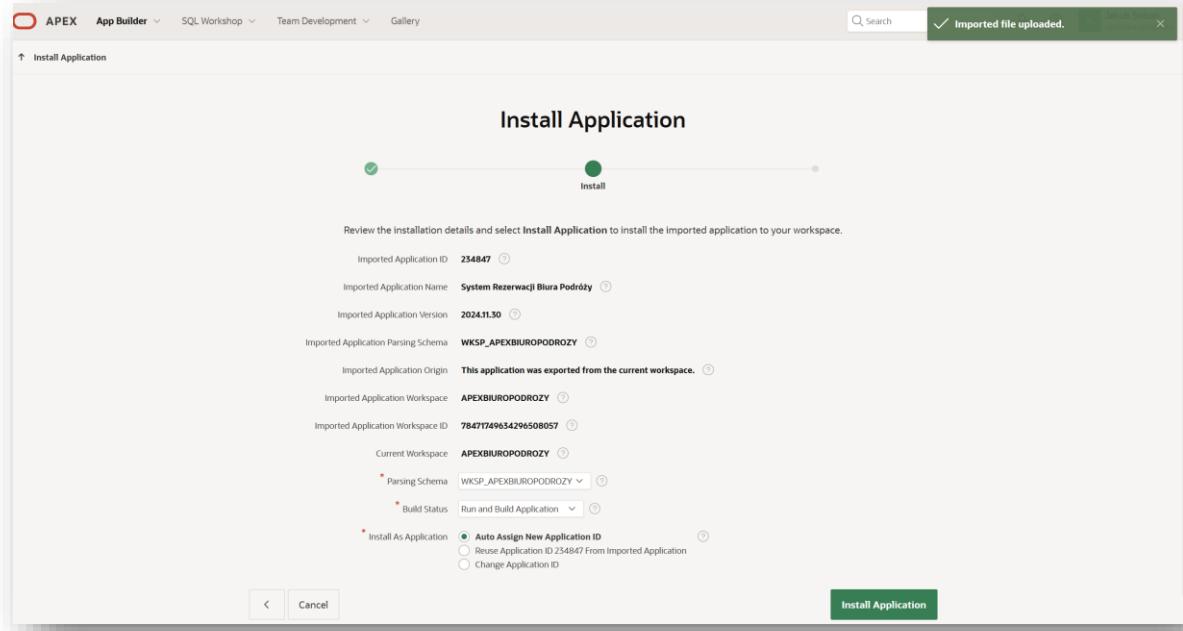
Parsing Schema: **WKSP_APEXBIUROPODROZY**

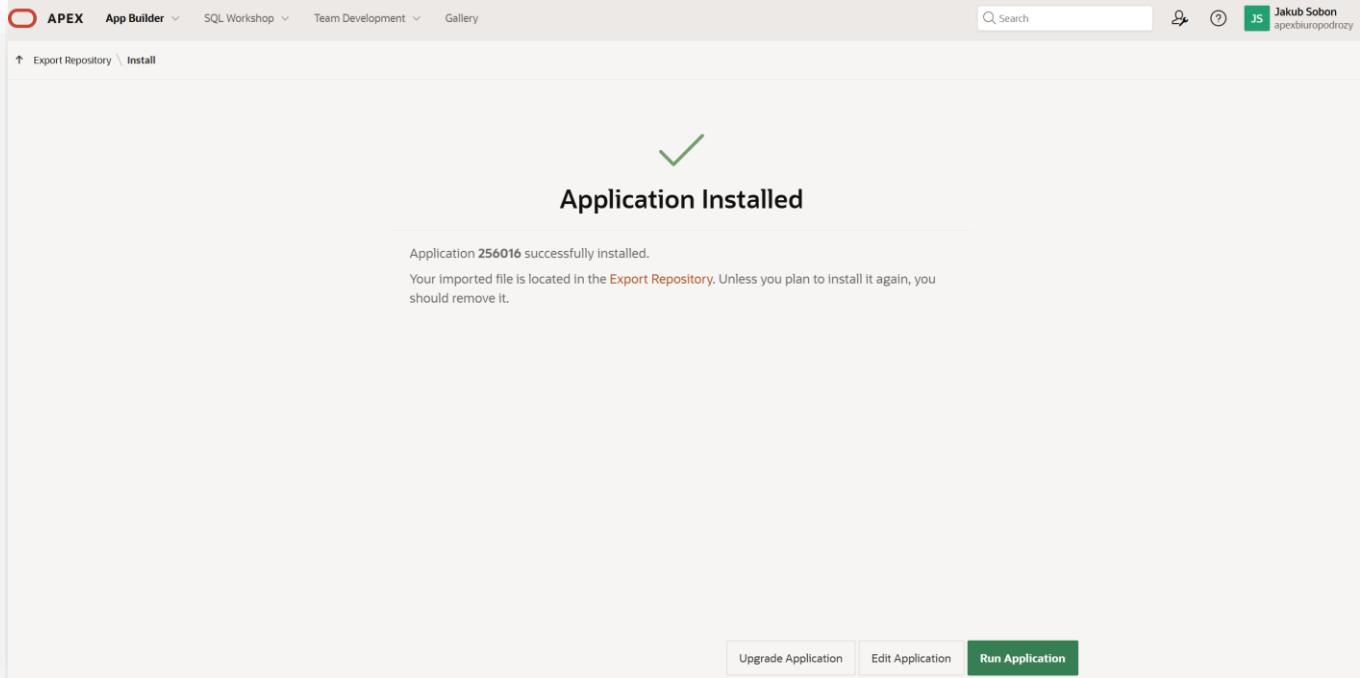
Build Status: **Run and Build Application**

Install As Application:

- Auto Assign New Application ID
- Reuse Application ID 234847 From Imported Application
- Change Application ID

< Cancel Install Application





Aby sprawdzić czy wszystko załadowało się poprawnie, dodam nowego klienta do bazy danych w nowym workspace:

The screenshot shows a page titled 'Usuń Klientów' (Delete Clients). At the top, there is a search bar labeled 'ID klienta' and a button labeled 'Usuń klinetę' (Delete client). Below this, there is a link 'Zarządzaj Klientami \'. The main area is titled 'Przeglądaj Klientami' (View Clients) and contains a table of client data. The table has columns: Nazwisko (Last Name), Imię (First Name), NIP (Social Security Number), Nazwa Firmy (Company Name), Adres (Address), Email, and Telefon (Phone). The data is as follows:

Nazwisko	Imię	NIP	Nazwa Firmy	Adres	Email	Telefon
Kowalski	Jan	1234567890	Firma Kowalski	1	jan.kowalski@email.com	9876543210
Nowak	Anna	2345678901	Firma Nowak	2	anna.nowak@email.com	9876543211
Wiśniewski	Piotr	3456789012	Firma Wiśniewski	3	piotr.wisniewski@email.com	9876543212
Janicki	Marek	4567890123	Firma Jankowski	4	marek.jankowski@email.com	9876543213
Lewandowski	Katarzyna	5678901234	Firma Lewandowski	5	katarzyna.lewandowski@email.com	9876543214
Zielinski	Adam	6789012345	Firma Zielinski	6	adam.zielinski@email.com	9876543215
Szymański	Ewa	7890123456	Firma Szymański	7	ewa.szymanski@email.com	9876543216
Wójcik	Tomasz	8901234567	Firma Wójcik	8	tomasz.wojcik@email.com	9876543217
Kowalczyk	Karolina	9012345678	Firma Kowalczyk	9	karolina.kowalczyk@email.com	9876543218
Kaczmarek	Michał	0123456789	Firma Kaczmarek	10	michal.kaczmarek@email.com	9876543219
Dąbrowski	Krzysztof	1122334455	Firma Dąbrowski	11	krzysztof.dabrowski@email.com	9876543220

Dodaj klienta

 Nazwisko
Test

 Imię
Próbyny

 NIP

 Firma

 Adres
Zbieżna 12

 Kod pocztowy
00-584

 Miasto
Warszawa

 Email
testowy@import.pl

 Telefon
543345053

 Dodaj klienta

 Pomyślnie dodano klienta
[Anuluj](#) [Dodaj klienta](#)

Nazwisko	Imię	Nip	Nazwa Firmy	Adres	Email	Telefon
Test	Próbyny			87	testowy@import.pl	543345053

6. Podsumowanie

W ramach projektu stworzono aplikację bazodanową w Oracle APEX dla biura podróży, która umożliwia kompleksowe zarządzanie bazą klientów, ofert turystycznych oraz rezerwacji. System został zaprojektowany z naciskiem na intuicyjną obsługę oraz efektywne przetwarzanie danych.

Aplikacja oferuje szeroki zakres funkcjonalności, takich jak:

- Dodawanie, edycja i usuwanie klientów,
- Zarządzanie ofertami turystycznymi, w tym ich dodawanie, modyfikacja, usuwanie oraz rezerwacje.

Projekt wykorzystuje technologie Oracle APEX i PL/SQL, które pozwalają na implementację logiki biznesowej, w tym procedur, funkcji oraz triggerów. Automatyzacja procesów, takich jak generowanie unikalnych numerów rezerwacji czy dynamiczne przydzielanie promocji, przyczynia się do zwiększenia wydajności i elastyczności systemu.

Podsumowując, aplikacja spełnia założenia projektowe, oferując niezawodność, bezpieczeństwo oraz łatwość użytkowania. Wykorzystanie nowoczesnych rozwiązań Oracle APEX i PL/SQL gwarantuje skalowalność oraz potencjał do dalszego rozwoju.