Programowanie proceduralne - widoki, procedury, triggery itp.

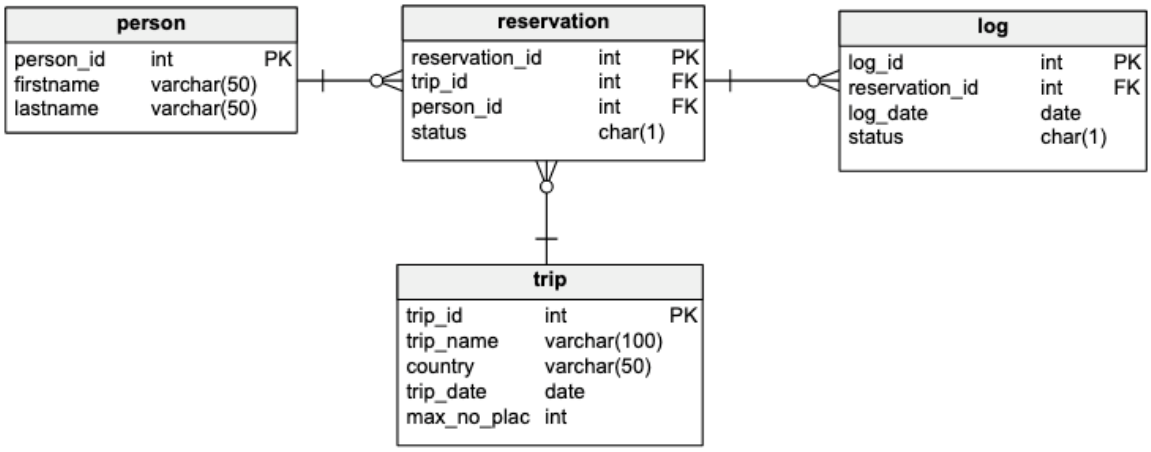
I. Oracle PL/SQL

1. Tabele

   Trip (trip_id, name, country, date, no_places)
   Person(person_id, firstname, lastname)
   Reservation(reservation_id, trip_id, person_id, status)

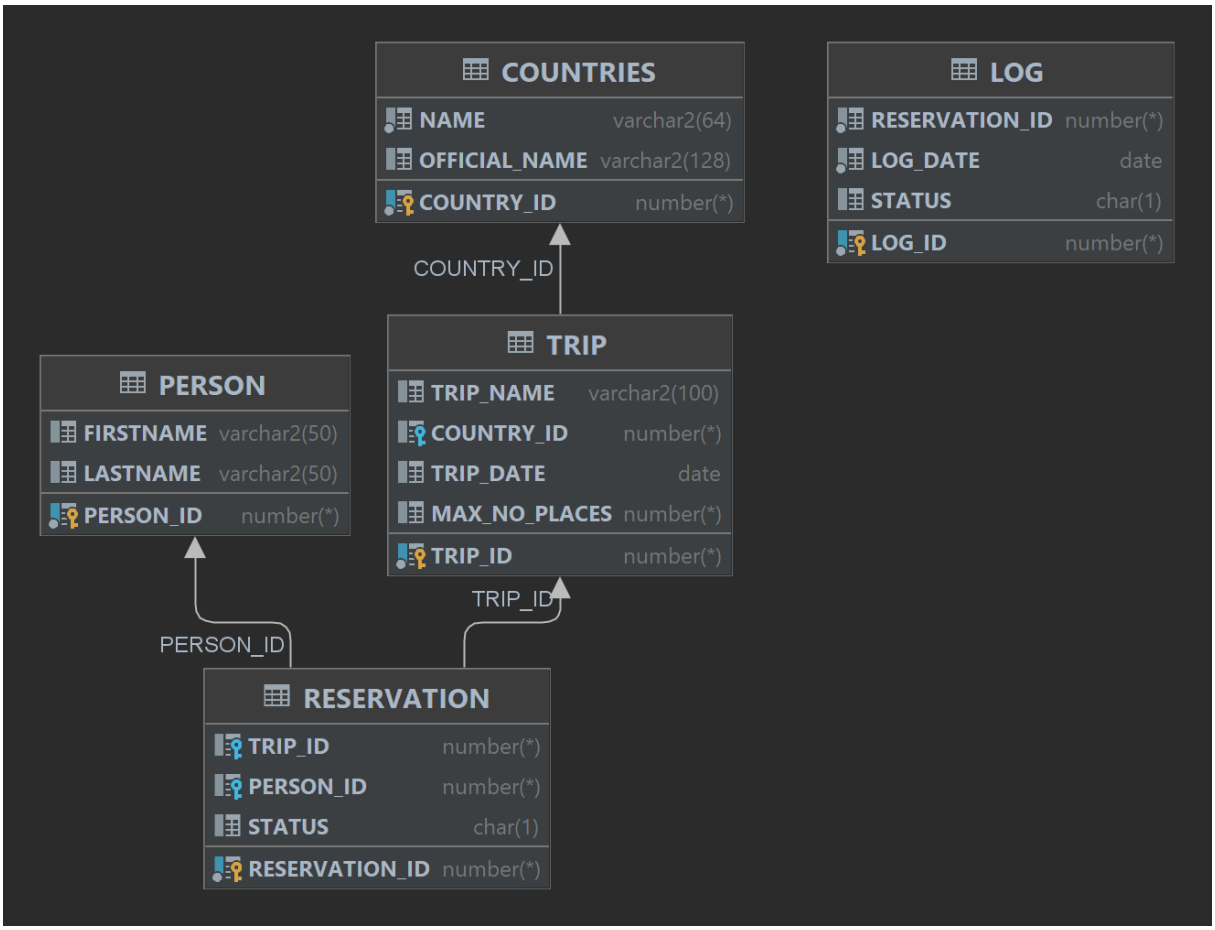
   Pole status w tabeli Rezerwacje może przyjmować jedną z 4 wartości

   N – New
   P – Paid
   C – Canceled



1a) Wypełnianie tabele przykładowymi danymi

Zmodyfikuj początkowy schemat bazy danych dodając tabelę "słownikową" z listą krajów



2. Wypełnianie tabele przykładowymi danymi
   4 wycieczki
   10 osób
   10 rezerwacji

   Dane testowe powinny być różnorodne (wycieczki w przyszłości, wycieczki w przeszłości, rezerwacje o różnym statusie itp.) tak, żeby umożliwić testowanie napisanych procedur.
   W razie potrzeby należy zmodyfikować dane tak żeby przetestować różne przypadki.

3. Tworzenie widoków. Należy przygotować kilka widoków ułatwiających dostęp do danych. Należy zwrócić uwagę na strukturę kodu (należy unikać powielania kodu)

   a) Reservations(country,trip_date,trip_name, firstname, lastname,reservation_id,status)

```
CREATE VIEW RESERVATIONS AS
SELECT
    COUNTRY_NAME,
    TRIP_DATE,
    TRIP_NAME,
    FIRSTNAME,
    LASTNAME,
    RESERVATION_ID,
    STATUS
FROM RESERVATIONS_
WITH READ ONLY;
```

```
+-------------+----------+-----------------+---------+----------+--------------+------+
|COUNTRY      |TRIP_DATE |TRIP_NAME        |FIRSTNAME|LASTNAME  |RESERVATION_ID|STATUS|
+-------------+----------+-----------------+---------+----------+--------------+------+
|France       |2022-09-12|Who was in Paris |Jan      |Nowak     |3             |P     |
|Poland       |2023-03-06|Foggy Kraków     |Jan      |Nowak     |5             |P     |
|Poland       |2023-12-30|Polish Hell      |Jan      |Nowak     |11            |N     |
|France       |2022-09-12|Who was in Paris |Jan      |Kowalski  |4             |N     |
|United States|2023-05-01|City of the Angels|Jan     |Nowakowski|10            |C     |
|Poland       |2023-12-30|Polish Hell      |Jan      |Nowakowski|12            |P     |
|Poland       |2023-03-06|Foggy Kraków     |Adam     |Kowalski  |6             |C     |
|United States|2023-05-01|City of the Angels|Piotr    |Piotrowski|7             |P     |
|United States|2023-05-01|City of the Angels|Piotr    |Piotrowski|8             |C     |
|United States|2023-05-01|City of the Angels|Piotr    |Piotrowski|9             |C     |
+-------------+----------+-----------------+---------+----------+--------------+------+
```

Widok pomocniczy:

```sql
CREATE VIEW RESERVATIONS_ AS
SELECT
    C.name,
    C.COUNTRY_ID,
    T.TRIP_ID,
    T.TRIP_DATE,
    T.TRIP_NAME,
    P.PERSON_ID,
    P.FIRSTNAME,
    P.LASTNAME,
    R.RESERVATION_ID,
    R.STATUS
FROM RESERVATION R
        INNER JOIN TRIP T ON R.TRIP_ID = T.TRIP_ID
        INNER JOIN COUNTRIES C ON T.COUNTRY_ID = C.COUNTRY_ID
        INNER JOIN PERSON P on R.PERSON_ID = P.PERSON_ID
WITH READ ONLY;
```

b) Trips(country,trip_date, trip_name,no_places, no_available_places)

```sql
CREATE VIEW TRIPS AS
SELECT COUNTRY_NAME, TRIP_DATE, TRIP_NAME, NO_PLACES, NO_AVAILABLE_PLACES FROM TRIPS_
WITH READ ONLY;
```

```
+-------------+----------+-----------------+---------+------------------+
|COUNTRY      |TRIP_DATE |TRIP_NAME        |NO_PLACES|NO_AVAILABLE_PLACES|
+-------------+----------+-----------------+---------+------------------+
|Poland       |2023-03-06|Foggy Kraków     |2        |1                 |
|Poland       |2023-12-30|Polish Hell      |32       |31                |
|United States|2023-05-01|City of the Angels|2       |1                 |
|France       |2022-09-12|Who was in Paris |2        |1                 |
+-------------+----------+-----------------+---------+------------------+
```

Widok pomocniczy:

```
CREATE VIEW TRIPS_ (
    TRIP_ID,
    COUNTRY_NAME,
    COUNTRY_ID,
    TRIP_DATE,
    TRIP_NAME,
    NO_PLACES,
    NO_AVAILABLE_PLACES
) AS
SELECT DISTINCT
    T.TRIP_ID,
    R.COUNTRY_NAME,
    R.COUNTRY_ID,
    T.TRIP_DATE,
    T.TRIP_NAME,
    T.MAX_NO_PLACES,
    T.MAX_NO_PLACES - TS.TRIPS_COUNT
FROM TRIP T
    INNER JOIN RESERVATIONS_ R ON T.TRIP_ID = R.TRIP_ID
    INNER JOIN (
        SELECT TRIP_ID, COUNT(*) TRIPS_COUNT
        FROM RESERVATIONS_ R
        WHERE R.STATUS = 'N' OR R.STATUS = 'P'
        GROUP BY R.TRIP_ID
    ) TS ON T.TRIP_ID = TS.TRIP_ID
WITH READ ONLY;
```

c) AvailableTrips(country,trip_date, trip_name,no_places, no_available_places)

```
CREATE VIEW AVAILABLETRIPS AS
SELECT COUNTRY_NAME, TRIP_DATE, TRIP_NAME, NO_PLACES, NO_AVAILABLE_PLACES FROM
VIEW_AVAILABLE_TRIPS_
WITH READ ONLY;
```

```
+-------------+----------+-----------------+---------+-------------------+
|COUNTRY      |TRIP_DATE |TRIP_NAME        |NO_PLACES|NO_AVAILABLE_PLACES|
+-------------+----------+-----------------+---------+-------------------+
|Poland       |2023-03-06|Foggy Kraków     |2        |1                  |
|Poland       |2023-12-30|Polish Hell      |32       |31                 |
|United States|2023-05-01|City of the Angels|2       |1                  |
|France       |2022-09-12|Who was in Paris |2        |1                  |
+-------------+----------+-----------------+---------+-------------------+
```

Widok pomocniczy:

```
CREATE OR REPLACE VIEW VIEW_AVAILABLE_TRIPS_ (COUNTRY, TRIP_ID, TRIP_DATE, TRIP_NAME,
NO_PLACES, NO_AVAILABLE_PLACES) AS

SELECT C.NAME, T.TRIP_ID, T.TRIP_DATE, T.TRIP_NAME, T.MAX_NO_PLACES, T.MAX_NO_PLACES -
TS.TRIPS_COUNT FROM TRIP T

    INNER JOIN COUNTRIES C on T.COUNTRY_ID = C.COUNTRY_ID
    INNER JOIN (
        SELECT TRIP_ID, COUNT(*) TRIPS_COUNT FROM RESERVATION WHERE STATUS = 'P' GROUP BY
TRIP_ID
    ) TS ON T.TRIP_ID = TS.TRIP_ID
    WHERE CURRENT_DATE <= T.TRIP_DATE
WITH READ ONLY;
```

Proponowany zestaw widoków można rozbudować wedle uznania/potrzeb
- np. można dodać nowe/pomocnicze widoki
- np. można zmienić def. widoków, dodając nowe/potrzebne pola

4. Tworzenie procedur/funkcji pobierających dane. Podobnie jak w poprzednim przykładzie należy przygotować kilka procedur ułatwiających dostęp do danych

   a) TripParticipants (trip_id), procedura ma zwracać podobny zestaw danych jak widok Reservations

```
CREATE TYPE TRIP_PARTICIPANT_OBJ AS OBJECT (
    FIRSTNAME           VARCHAR(50),
    LASTNAME            VARCHAR(50),
    TRIP_NAME           VARCHAR(100),
    TRIP_DATE           DATE,
    COUNTRY_NAME        VARCHAR(64),
    RESERVATION_ID      INT,
    RESERVATION_STATUS  VARCHAR(1)
);

CREATE TYPE TRIP_PARTICIPANTS_TABLE AS TABLE OF TRIP_PARTICIPANT_OBJ;
```

```
CREATE FUNCTION TripParticipants(trip_id INT)
RETURN TRIP_PARTICIPANTS_TABLE
AS
    l_result TRIP_PARTICIPANTS_TABLE;

    l_trip_number INT;
    l_trip_id TRIP.TRIP_ID%TYPE := trip_id;
BEGIN
    SELECT COUNT(*) INTO l_trip_number FROM TRIP WHERE TRIP.TRIP_ID = l_trip_id;

    IF l_trip_number = 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Trip ' || trip_id || ' does not exist');
    END IF;

    SELECT TRIP_PARTICIPANT_OBJ(
        P.FIRSTNAME,
        P.LASTNAME,
        T.TRIP_NAME,
        T.TRIP_DATE,
        C.name,
        R.RESERVATION_ID,
        R.STATUS
    ) BULK COLLECT INTO l_result
    FROM RESERVATION R
    INNER JOIN TRIP T ON R.TRIP_ID = T.TRIP_ID
    INNER JOIN COUNTRIES C ON T.COUNTRY_ID = C.COUNTRY_ID
    INNER JOIN PERSON P on R.PERSON_ID = P.PERSON_ID
        WHERE T.TRIP_ID = l_trip_id;

    RETURN l_result;
END;
```

b) PersonReservations(person_id), procedura ma zwracać podobny zestaw danych jak widok Reservations

```
CREATE FUNCTION PersonReservations(person_id INT)
RETURN TRIP_PARTICIPANTS_TABLE
AS
    l_result TRIP_PARTICIPANTS_TABLE;

    l_person_number INT;
    l_person_id TRIP.TRIP_ID%TYPE := person_id;
BEGIN
    SELECT COUNT(*) INTO l_person_number FROM TRIP;

    IF l_person_number = 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Person ' || person_id || ' does not exist');
    END IF;

    SELECT TRIP_PARTICIPANT_OBJ(
        P.FIRSTNAME,
        P.LASTNAME,
        T.TRIP_NAME,
        T.TRIP_DATE,
        C.name,
        R.RESERVATION_ID,
        R.STATUS
    ) BULK COLLECT INTO l_result
    FROM RESERVATION R
    INNER JOIN TRIP T ON R.TRIP_ID = T.TRIP_ID
    INNER JOIN COUNTRIES C ON T.COUNTRY_ID = C.COUNTRY_ID
    INNER JOIN PERSON P on R.PERSON_ID = P.PERSON_ID
        WHERE P.PERSON_ID = l_person_id;

    RETURN l_result;
END;
```

c) AvailableTrips(country, date_from, date_to)

```
CREATE TYPE TRIP_OBJ AS OBJECT (
    TRIP_NAME VARCHAR(100),
    TRIP_DATE DATE,
    COUNTRY_NAME VARCHAR(128),
    MAX_NO_PLACES INT,
    AVAILABLE_NO_PLACES INT
);

CREATE TYPE TRIP_TABLE AS TABLE OF TRIP_OBJ;
```

```
CREATE FUNCTION Fn_AvailableTrips(country_name VARCHAR, date_from DATE, date_to DATE)
RETURN TRIP_TABLE
AS
    l_result TRIP_TABLE;

    l_country_count INT;
    l_date_diff INT := date_to - date_from;
BEGIN
    SELECT COUNT(*) INTO l_country_count FROM COUNTRIES WHERE NAME = country_name;

    IF l_country_count = 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Country ' || country_name || ' does not exist');
    END IF;
    IF l_date_diff < 0 THEN
        RAISE_APPLICATION_ERROR(-20002, 'Wrong dates');
    END IF;


    SELECT TRIP_OBJ(
        T.TRIP_NAME,
        T.TRIP_DATE,
        T.COUNTRY_NAME,
        T.NO_PLACES,
        T.NO_AVAILABLE_PLACES
    ) BULK COLLECT INTO l_result
    FROM AVAILABLETRIPS T
    WHERE
        T.COUNTRY_NAME = Fn_AvailableTrips.country_name
        AND T.TRIP_DATE BETWEEN date_from AND date_to;

    RETURN l_result;
END;
```

Procedury/funkcje powinny zwracać tabelę/zbiór wynikowy

Należy zwrócić uwagę na kontrolę parametrów (np. jeśli parametrem jest trip_id to należy sprawdzić czy taka wycieczka istnieje). Podobnie jak w przypadku widoków należy unikać powielania kodu.

5. Tworzenie procedur modyfikujących dane. Należy przygotować zestaw procedur pozwalających na modyfikację danych oraz kontrolę poprawności ich wprowadzania

   a) AddReservation(trip_id, person_id), procedura powinna kontrolować czy wycieczka jeszcze się nie odbyła, i czy sa wolne miejsca

```
CREATE PROCEDURE AddReservation(trip_id INT, person_id INT)
AS
    l_trip_id INT := trip_id;
    l_person_id INT := person_id;
    l_available_no_places INT;
    l_trip_number INT;
    l_person_number INT;
BEGIN
    SELECT COUNT(*) INTO l_trip_number FROM TRIP WHERE TRIP_ID = l_trip_id;
    IF l_trip_number = 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Trip ' || trip_id || ' does not exist');
    END IF;

    SELECT COUNT(*) INTO l_person_number FROM PERSON WHERE PERSON_ID = l_person_id;
    IF l_person_number = 0 THEN
        RAISE_APPLICATION_ERROR(-20002, 'Person ' || person_id || ' does not exist');
    END IF;

    SELECT NO_AVAILABLE_PLACES INTO l_available_no_places FROM AVAILABLETRIPS;
    IF l_available_no_places <= 0 THEN
        RAISE_APPLICATION_ERROR(-20003, 'Trip ' || trip_id || ' does not have any free
spots');
    END IF;

    INSERT INTO RESERVATION (TRIP_ID, PERSON_ID, STATUS)
    VALUES (l_trip_id, l_person_id, 'N');
END;
```

b) ModifyReservationStatus(reservation_id, status), procedura kontrolować czy możliwa jest zmiana statusu, np. zmiana statusu już anulowanej wycieczki (przywrócenie do stanu aktywnego nie zawsze jest możliwa – może już nie być miejsc)

```
CREATE PROCEDURE ModifyReservationStatus(reservation_id INT, status VARCHAR)
AS
    l_reservation_id INT := reservation_id;
    l_status VARCHAR(1) := status;
    l_trip_id TRIP.TRIP_ID%TYPE;
    l_reservation_count INT;
    l_current_status VARCHAR(1);
    l_available_no_places INT;

BEGIN
    SELECT COUNT(*) INTO l_reservation_count FROM RESERVATION R WHERE R.RESERVATION_ID =
l_reservation_id;
    IF l_reservation_count = 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Reservation ' || reservation_id || ' does not
exist');
    END IF;

    SELECT R.STATUS INTO l_current_status FROM RESERVATION R WHERE R.RESERVATION_ID =
l_reservation_id;
    IF l_status = l_current_status THEN
        RAISE_APPLICATION_ERROR(-20002, 'Current reservation ' || reservation_id || ' status
is equal to passed argument');
    END IF;

    SELECT TRIP_ID INTO l_trip_id
    FROM RESERVATION R
    WHERE R.RESERVATION_ID = l_reservation_id;

    IF status = 'N' OR status = 'P' THEN
        SELECT NO_AVAILABLE_PLACES INTO l_available_no_places
        FROM VIEW_AVAILABLE_TRIPS_
        WHERE TRIP_ID = l_trip_id;

        IF l_available_no_places = 0 THEN
            RAISE_APPLICATION_ERROR(-20003, 'Trip ' || l_trip_id || ' does not have any more
free spots');
        END IF;
    END IF;

    UPDATE RESERVATION
    SET STATUS = l_status
    WHERE RESERVATION_ID = l_reservation_id;
END;
```

c) ModifyNoPlaces(trip_id, no_places), nie wszystkie zmiany liczby miejsc są dozwolone, nie można zmniejszyć liczby miejsc na wartość poniżej liczby zarezerwowanych miejsc

```
CREATE PROCEDURE ModifyNoPlaces(trip_id INT, no_places INT) AS
    l_trip_id INT := trip_id;
    l_trip_count INT;
    l_trip_available_no_places INT;
BEGIN
    SELECT COUNT(*) INTO l_trip_count
    FROM VIEW_AVAILABLE_TRIPS_ T
    WHERE T.TRIP_ID = l_trip_id;
    IF l_trip_count = 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Trip ' || trip_id || ' does not exist or is not
available');
    END IF;

    SELECT NO_AVAILABLE_PLACES INTO l_trip_available_no_places
    FROM VIEW_AVAILABLE_TRIPS_ T
    WHERE T.TRIP_ID = l_trip_id;
    IF l_trip_available_no_places - no_places < 0 THEN
        RAISE_APPLICATION_ERROR(-20002, 'There are not free slots enough');
    END IF;

    UPDATE TRIP
    SET MAX_NO_PLACES = no_places
    WHERE TRIP_ID = l_trip_id;
END;
```

Należy rozważyć użycie transakcji
Należy zwrócić uwagę na kontrolę parametrów (np. jeśli parametrem jest trip_id to należy sprawdzić czy taka wycieczka istnieje, jeśli robimy rezerwację to należy sprawdzać czy są wolne miejsca itp..)

Nie skorzystałem z transakcji, ponieważ wszystkie przypadki kontrolują odpowiednie instrukcje warunkowe, które rzucają wyjątki za pomocą `RAISE_APPLICATION_ERROR()`.

6. Dodajemy tabelę dziennikującą zmiany statusu rezerwacji
   Log(id, reservation_id, date, status)

   Należy zmienić warstwę procedur modyfikujących dane tak aby dopisywały informację do dziennika

```
CREATE PROCEDURE ModifyReservationStatus(reservation_id INT, status VARCHAR)
AS
    l_reservation_id INT := reservation_id;
    l_status VARCHAR(1) := status;
    l_trip_id TRIP.TRIP_ID%TYPE;
    l_reservation_count INT;
    l_current_status VARCHAR(1);
    l_available_no_places INT;

BEGIN
    SELECT COUNT(*) INTO l_reservation_count FROM RESERVATION R WHERE R.RESERVATION_ID =
l_reservation_id;
    IF l_reservation_count = 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Reservation ' || reservation_id || ' does not exist');
    END IF;

    SELECT R.STATUS INTO l_current_status FROM RESERVATION R WHERE R.RESERVATION_ID =
l_reservation_id;
    IF l_status = l_current_status THEN
        RAISE_APPLICATION_ERROR(-20002, 'Current reservation ' || reservation_id || ' status is
equal to passed argument');
    END IF;

    SELECT TRIP_ID INTO l_trip_id
    FROM RESERVATION R
    WHERE R.RESERVATION_ID = l_reservation_id;

    IF status = 'N' OR status = 'P' THEN
        SELECT NO_AVAILABLE_PLACES INTO l_available_no_places
        FROM VIEW_AVAILABLE_TRIPS_
        WHERE TRIP_ID = l_trip_id;

        IF l_available_no_places = 0 THEN
            RAISE_APPLICATION_ERROR(-20003, 'Trip ' || l_trip_id || ' does not have any more
free spots');
        END IF;
    END IF;

    INSERT INTO LOG (RESERVATION_ID, LOG_DATE, STATUS)
    VALUES (RESERVATION_ID, CURRENT_DATE, l_current_status);

    UPDATE RESERVATION
    SET STATUS = l_status
    WHERE RESERVATION_ID = l_reservation_id;
END;
```

7. Zmiana strategii zapisywania do dziennika rezerwacji. Realizacja przy pomocy triggerów

   Należy wprowadzić zmianę, która spowoduje, że zapis do dziennika rezerwacji będzie realizowany przy pomocy trierów

   a) Trigger obsługujący dodanie rezerwacji

```
CREATE OR REPLACE TRIGGER TR_ON_ADD_RESERVATION
    AFTER INSERT
    ON RESERVATION
    FOR EACH ROW
BEGIN
    INSERT INTO LOG (RESERVATION_ID, LOG_DATE, STATUS)
    VALUES (RESERVATION_ID, CURRENT_DATE, :NEW.STATUS);
END;

CREATE OR REPLACE PROCEDURE ADDRESERVATION2(trip_id INT, person_id INT)
AS
    l_trip_id INT := trip_id;
    l_person_id INT := person_id;
    l_available_no_places INT;
    l_trip_number INT;
    l_person_number INT;
BEGIN
    SELECT COUNT(*) INTO l_trip_number FROM TRIP T WHERE T.TRIP_ID = l_trip_id;
    IF l_trip_number = 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Trip ' || trip_id || ' does not exist');
    END IF;

    SELECT COUNT(*) INTO l_person_number FROM PERSON P WHERE P.PERSON_ID = l_person_id;
    IF l_person_number = 0 THEN
        RAISE_APPLICATION_ERROR(-20002, 'Person ' || person_id || ' does not exist');
    END IF;

    SELECT NO_AVAILABLE_PLACES INTO l_available_no_places FROM AVAILABLETRIPS;
    IF l_available_no_places <= 0 THEN
        RAISE_APPLICATION_ERROR(-20003, 'Trip ' || trip_id || ' does not have any free
spots');
    END IF;

    INSERT INTO RESERVATION (TRIP_ID, PERSON_ID, STATUS)
    VALUES (l_trip_id, l_person_id, 'N');
END;
```

b) Trigger obsługujący zmianę statusu

```
CREATE OR REPLACE TRIGGER TR_ON_MODIFY_RESERVATION_STATUS
    AFTER UPDATE
    ON RESERVATION
    FOR EACH ROW
DECLARE
BEGIN
    INSERT INTO LOG (RESERVATION_ID, LOG_DATE, STATUS)
    VALUES (RESERVATION_ID, CURRENT_DATE, :OLD.STATUS);
END;

CREATE OR REPLACE PROCEDURE ModifyReservationStatus2(reservation_id INT, status VARCHAR)
AS
    l_reservation_id INT := reservation_id;
    l_status VARCHAR(1) := status;
    l_trip_id TRIP.TRIP_ID%TYPE;
    l_reservation_count INT;
    l_current_status VARCHAR(1);
    l_available_no_places INT;

BEGIN
    SELECT COUNT(*) INTO l_reservation_count FROM RESERVATION R WHERE R.RESERVATION_ID =
l_reservation_id;
    IF l_reservation_count = 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Reservation ' || reservation_id || ' does not
exist');
    END IF;

    SELECT R.STATUS INTO l_current_status FROM RESERVATION R WHERE R.RESERVATION_ID =
l_reservation_id;
    IF l_status = l_current_status THEN
        RAISE_APPLICATION_ERROR(-20002, 'Current reservation ' || reservation_id || ' status
is equal to passed argument');
    END IF;

    SELECT TRIP_ID INTO l_trip_id
    FROM RESERVATION R
    WHERE R.RESERVATION_ID = l_reservation_id;

    IF status = 'N' OR status = 'P' THEN
        SELECT NO_AVAILABLE_PLACES INTO l_available_no_places
        FROM VIEW_AVAILABLE_TRIPS_
        WHERE TRIP_ID = l_trip_id;

        IF l_available_no_places = 0 THEN
            RAISE_APPLICATION_ERROR(-20003, 'Trip ' || l_trip_id || ' does not have any more
free spots');
        END IF;
    END IF;

    UPDATE RESERVATION
    SET STATUS = l_status
    WHERE RESERVATION_ID = l_reservation_id;
END;
```

c) Trigger zabraniający usunięcia rezerwacji

```
CREATE OR REPLACE TRIGGER TR_ON_DELETE_RESERVATION
    BEFORE DELETE ON RESERVATION
    FOR EACH ROW
BEGIN
    RAISE_APPLICATION_ERROR(-20001, 'Cannot delete reservation records');
END;
```

Oczywiście po wprowadzeniu tej zmiany należy uaktualnić procedury modyfikujące dane. Najlepiej to zrobić tworząc nowe wersje (np. z sufiksem 2)

8. Zmiana strategii kontroli dostępności miejsc. Realizacja przy pomocy triggerów

   Należy wprowadzić zmianę, która spowoduje, że zapis do dziennika rezerwacji będzie realizowany przy pomocy trierów

   a) Trigger obsługujący dodanie rezerwacji

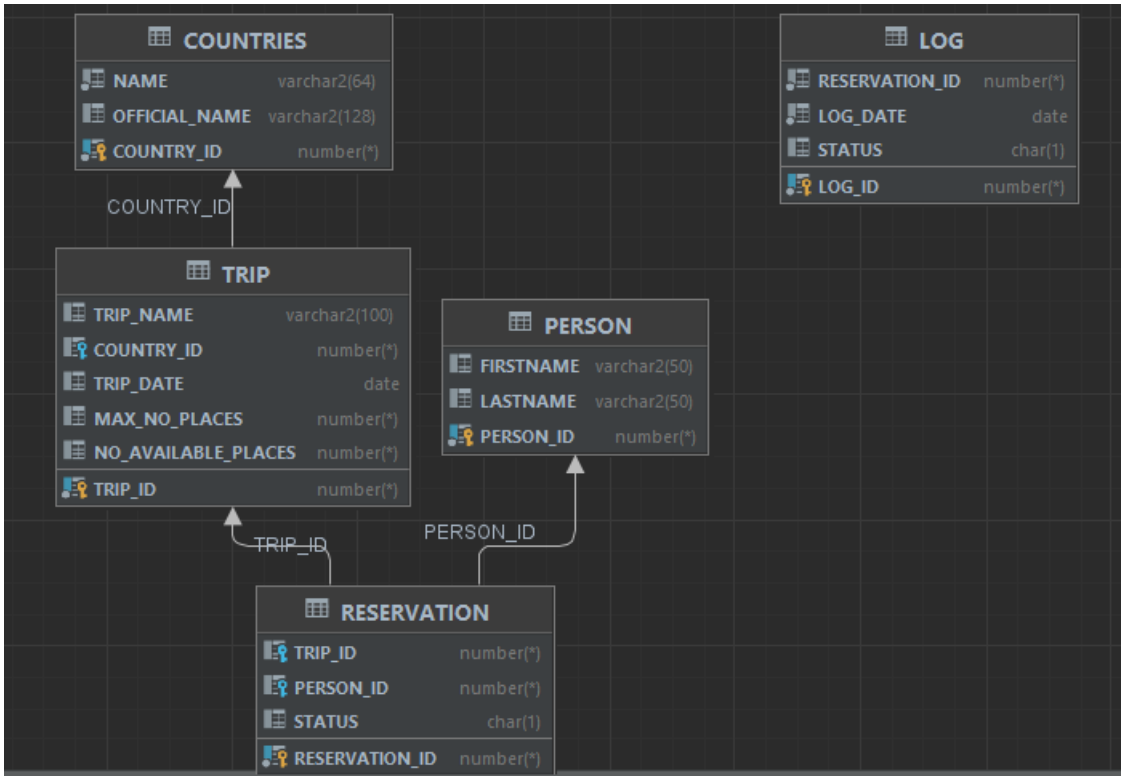
```
CREATE OR REPLACE TRIGGER TR_ON_ADD_RESERVATION
    AFTER INSERT
    ON RESERVATION
    FOR EACH ROW
DECLARE
    l_available_no_places INT;
BEGIN
    SELECT NO_AVAILABLE_PLACES INTO l_available_no_places FROM AVAILABLETRIPS;
    IF l_available_no_places <= 0 THEN
        RAISE_APPLICATION_ERROR(-20003, 'Trip ' || :new.TRIP_ID || ' does not have any free
spots');
    END IF;

    INSERT INTO LOG (RESERVATION_ID, LOG_DATE, STATUS)
    VALUES (RESERVATION_ID, CURRENT_DATE, :new.STATUS);
END;

CREATE OR REPLACE PROCEDURE ADDRESERVATION3(trip_id INT, person_id INT)
AS
    l_trip_id INT := trip_id;
    l_person_id INT := person_id;
    l_trip_number INT;
    l_person_number INT;
BEGIN
    SELECT COUNT(*) INTO l_trip_number FROM TRIP T WHERE T.TRIP_ID = l_trip_id;
    IF l_trip_number = 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Trip ' || trip_id || ' does not exist');
    END IF;

    SELECT COUNT(*) INTO l_person_number FROM PERSON P WHERE P.PERSON_ID = l_person_id;
    IF l_person_number = 0 THEN
        RAISE_APPLICATION_ERROR(-20002, 'Person ' || person_id || ' does not exist');
    END IF;

    INSERT INTO RESERVATION (TRIP_ID, PERSON_ID, STATUS)
    VALUES (l_trip_id, l_person_id, 'N');
END;
```

b) Trigger obsługujący zmianę statusu

```
CREATE OR REPLACE TRIGGER TR_ON_MODIFY_RESERVATION_STATUS
    AFTER UPDATE
    ON RESERVATION
    FOR EACH ROW
DECLARE
    l_available_no_places INT;
BEGIN
    IF :NEW.STATUS = 'N' OR :NEW.STATUS = 'P' THEN
        SELECT NO_AVAILABLE_PLACES INTO l_available_no_places
        FROM VIEW_AVAILABLE_TRIPS_ T
        WHERE T.TRIP_ID = :NEW.TRIP_ID;

        IF l_available_no_places = 0 THEN
            RAISE_APPLICATION_ERROR(-20003, 'Trip ' || :NEW.TRIP_ID || ' does not have any
more free spots');
        END IF;
    END IF;

    INSERT INTO LOG (RESERVATION_ID, LOG_DATE, STATUS)
    VALUES (RESERVATION_ID, CURRENT_DATE, :OLD.STATUS);
END;

CREATE OR REPLACE PROCEDURE ModifyReservationStatus3(reservation_id INT, status VARCHAR)
AS
    l_reservation_id INT := reservation_id;
    l_status VARCHAR(1) := status;
    l_reservation_count INT;
    l_current_status VARCHAR(1);
BEGIN
    SELECT COUNT(*) INTO l_reservation_count FROM RESERVATION R WHERE R.RESERVATION_ID =
l_reservation_id;
    IF l_reservation_count = 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Reservation ' || reservation_id || ' does not
exist');
    END IF;

    SELECT R.STATUS INTO l_current_status FROM RESERVATION R WHERE R.RESERVATION_ID =
l_reservation_id;
    IF l_status = l_current_status THEN
        RAISE_APPLICATION_ERROR(-20002, 'Current reservation ' || reservation_id || ' status
is equal to passed argument');
    END IF;

    UPDATE RESERVATION
    SET STATUS = l_status
    WHERE RESERVATION_ID = l_reservation_id;
END;
```

Oczywiście po wprowadzeniu tej zmiany należy uaktualnić procedury modyfikujące dane. Najlepiej to zrobić tworząc nowe wersje (np. z sufiksem 3)

9. Zmiana struktury bazy danych, w tabeli wycieczki dodajemy redundantne pole no_available_places



Obsługa redundantnego pola (kontrola liczby dostępnych miejsc) w procedurach

```
CREATE OR REPLACE PROCEDURE ADDRESERVATION4(trip_id INT, person_id INT)
AS
    l_trip_id INT := trip_id;
    l_person_id INT := person_id;
    l_trip_number INT;
    l_person_number INT;
BEGIN
    SELECT COUNT(*) INTO l_trip_number FROM TRIP T WHERE T.TRIP_ID = l_trip_id;
    IF l_trip_number = 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Trip ' || trip_id || ' does not exist');
    END IF;

    SELECT COUNT(*) INTO l_person_number FROM PERSON P WHERE P.PERSON_ID = l_person_id;
    IF l_person_number = 0 THEN
        RAISE_APPLICATION_ERROR(-20002, 'Person ' || person_id || ' does not exist');
    END IF;

    INSERT INTO RESERVATION (TRIP_ID, PERSON_ID, STATUS)
    VALUES (l_trip_id, l_person_id, 'N');

    UPDATE TRIP
    SET NO_AVAILABLE_PLACES = NO_AVAILABLE_PLACES - 1
    WHERE TRIP_ID = l_trip_id;
END;

CREATE OR REPLACE PROCEDURE ModifyReservationStatus4(reservation_id INT, status VARCHAR)
AS
    l_reservation_id INT := reservation_id;
    l_status VARCHAR(1) := status;
    l_reservation_count INT;
    l_current_status VARCHAR(1);
    l_trip_id INT;
BEGIN
    SELECT COUNT(*) INTO l_reservation_count FROM RESERVATION R WHERE R.RESERVATION_ID =
l_reservation_id;
    IF l_reservation_count = 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Reservation ' || reservation_id || ' does not exist');
    END IF;

    SELECT R.STATUS INTO l_current_status FROM RESERVATION R WHERE R.RESERVATION_ID =
l_reservation_id;
    IF l_status = l_current_status THEN
        RAISE_APPLICATION_ERROR(-20002, 'Current reservation ' || reservation_id || ' status is
equal to passed argument');
    END IF;

    IF status IN ('N', 'P') THEN
        SELECT TRIP_ID INTO l_trip_id
        FROM RESERVATION R
        WHERE R.RESERVATION_ID = l_reservation_id;

        UPDATE TRIP
        SET NO_AVAILABLE_PLACES = NO_AVAILABLE_PLACES - 1
        WHERE TRIP_ID = l_trip_id;
    END IF;

    UPDATE RESERVATION
```

```
    SET STATUS = l_status
    WHERE RESERVATION_ID = l_reservation_id;
END;
```

Należy zmodyfikować zestaw widoków. Proponuję dodać kolejne widoki (np. z sufiksem 4), które pobierają informację o wolnych miejscach z nowo dodanego pola.

```
CREATE OR REPLACE VIEW VIEW_AVAILABLE_TRIPS_ (COUNTRY, TRIP_ID, TRIP_DATE, TRIP_NAME, NO_PLACES,
NO_AVAILABLE_PLACES) AS
SELECT C.NAME, T.TRIP_ID, T.TRIP_DATE, T.TRIP_NAME, T.MAX_NO_PLACES, T.NO_AVAILABLE_PLACES FROM
TRIP T
    INNER JOIN COUNTRIES C on T.COUNTRY_ID = C.COUNTRY_ID
    WHERE CURRENT_DATE <= T.TRIP_DATE AND T.NO_AVAILABLE_PLACES > 0
WITH READ ONLY;

CREATE OR REPLACE VIEW TRIPS_ (TRIP_ID, COUNTRY_NAME, COUNTRY_ID, TRIP_DATE, TRIP_NAME,
NO_PLACES, NO_AVAILABLE_PLACES) as
SELECT DISTINCT
    T.TRIP_ID,
    R.COUNTRY_NAME,
    R.COUNTRY_ID,
    T.TRIP_DATE,
    T.TRIP_NAME,
    T.MAX_NO_PLACES,
    T.NO_AVAILABLE_PLACES
FROM TRIP T
    INNER JOIN RESERVATIONS_ R ON T.TRIP_ID = R.TRIP_ID
WITH READ ONLY;
```

Należy napisać procedurę przelicz która zaktualizuje wartość liczby wolnych miejsc dla już istniejących danych

```
CREATE OR REPLACE PROCEDURE PR_UPDATE_AVAILABLE_NO_PLACES AS
    l_trip_count INT;
BEGIN
    SELECT COUNT(*) INTO l_trip_count FROM TRIP T;

    FOR row IN (SELECT * FROM TRIP T) LOOP
        UPDATE TRIP T
        SET T.NO_AVAILABLE_PLACES = (SELECT MAX_NO_PLACES FROM TRIP) - (
            SELECT COUNT(*)
            FROM RESERVATIONS_ R
            WHERE R.STATUS IN ('N', 'P') AND R.TRIP_ID = row.TRIP_ID
        )
        WHERE T.TRIP_ID = row.TRIP_ID;
    END LOOP;
END;
```

Należy zmodyfikować warstwę procedur/funkcji pobierających dane, podobnie jak w przypadku widoków.

Należy zmodyfikować procedury wprowadzające dane tak aby korzystały/aktualizowały pole no_available_places w tabeli wycieczki
Najlepiej to zrobić tworząc nowe wersje (np. z sufiksem 4)

10. Zmiana strategii obsługi redundantnego pola no_available_places, realizacja przy pomocy trigerów

a) Trigger obsługujący dodanie rezerwacji

```
CREATE OR REPLACE TR_ON_ADD_RESERVATION
    after insert
    on RESERVATION
    for each row
DECLARE
    l_available_no_places INT;
BEGIN
    SELECT NO_AVAILABLE_PLACES INTO l_available_no_places FROM AVAILABLETRIPS;
    IF l_available_no_places <= 0 THEN
        RAISE_APPLICATION_ERROR(-20003, 'Trip ' || :new.TRIP_ID || ' does not have any free
spots');
    END IF;

    INSERT INTO LOG (RESERVATION_ID, LOG_DATE, STATUS)
    VALUES (RESERVATION_ID, CURRENT_DATE, :new.STATUS);

    UPDATE TRIP
    SET NO_AVAILABLE_PLACES = NO_AVAILABLE_PLACES - 1
    WHERE TRIP_ID = :NEW.TRIP_ID;
END;

CREATE OR REPLACE PROCEDURE ADDRESERVATION5(trip_id INT, person_id INT)
AS
    l_trip_id INT := trip_id;
    l_person_id INT := person_id;
    l_trip_number INT;
    l_person_number INT;
BEGIN
    SELECT COUNT(*) INTO l_trip_number FROM TRIP T WHERE T.TRIP_ID = l_trip_id;
    IF l_trip_number = 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Trip ' || trip_id || ' does not exist');
    END IF;

    SELECT COUNT(*) INTO l_person_number FROM PERSON P WHERE P.PERSON_ID = l_person_id;
    IF l_person_number = 0 THEN
        RAISE_APPLICATION_ERROR(-20002, 'Person ' || person_id || ' does not exist');
    END IF;

    INSERT INTO RESERVATION (TRIP_ID, PERSON_ID, STATUS)
    VALUES (l_trip_id, l_person_id, 'N');
END;
```

b) Trigger obsługujący zmianę statusu

```
CREATE OR REPLACE TRIGGER TR_ON_MODIFY_RESERVATION_STATUS
    AFTER UPDATE
    ON RESERVATION
    FOR EACH ROW
DECLARE
    l_available_no_places INT;
BEGIN
    IF :NEW.STATUS = 'N' OR :NEW.STATUS = 'P' THEN
        SELECT NO_AVAILABLE_PLACES INTO l_available_no_places
        FROM VIEW_AVAILABLE_TRIPS_ T
        WHERE T.TRIP_ID = :NEW.TRIP_ID;

        IF l_available_no_places = 0 THEN
            RAISE_APPLICATION_ERROR(-20003, 'Trip ' || :NEW.TRIP_ID || ' does not have any
more free spots');
        END IF;

        UPDATE TRIP
        SET NO_AVAILABLE_PLACES = NO_AVAILABLE_PLACES - 1
        WHERE TRIP_ID = :NEW.TRIP_ID;
    END IF;

    INSERT INTO LOG (RESERVATION_ID, LOG_DATE, STATUS)
    VALUES (RESERVATION_ID, CURRENT_DATE, :OLD.STATUS);
END;

CREATE OR REPLACE PROCEDURE ModifyReservationStatus5(reservation_id INT, status VARCHAR)
AS
    l_reservation_id INT := reservation_id;
    l_status VARCHAR(1) := status;
    l_reservation_count INT;
    l_current_status VARCHAR(1);
    l_trip_id INT;
BEGIN
    SELECT COUNT(*) INTO l_reservation_count FROM RESERVATION R WHERE R.RESERVATION_ID =
l_reservation_id;
    IF l_reservation_count = 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Reservation ' || reservation_id || ' does not
exist');
    END IF;

    SELECT R.STATUS INTO l_current_status FROM RESERVATION R WHERE R.RESERVATION_ID =
l_reservation_id;
    IF l_status = l_current_status THEN
        RAISE_APPLICATION_ERROR(-20002, 'Current reservation ' || reservation_id || ' status
is equal to passed argument');
    END IF;

    IF status IN ('N', 'P') THEN

    END IF;

    UPDATE RESERVATION
    SET STATUS = l_status
    WHERE RESERVATION_ID = l_reservation_id;
END;
```

c) Trigger obsługujący zmianę liczby miejsc na poziomie wycieczki

```
CREATE OR REPLACE TRIGGER TR_ON_TRIP_UPDATE
    BEFORE UPDATE
    ON TRIP
    FOR EACH ROW
DECLARE
    l_trip_count INT;
    l_trip_available_no_places INT;
BEGIN
    SELECT COUNT(*) INTO l_trip_count
    FROM VIEW_AVAILABLE_TRIPS_ T
    WHERE T.TRIP_ID = :NEW.TRIP_ID;
    IF l_trip_count = 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Trip ' || :NEW.TRIP_ID || ' does not exist or is
not available');
    END IF;

    SELECT NO_AVAILABLE_PLACES INTO l_trip_available_no_places
    FROM VIEW_AVAILABLE_TRIPS_ T
    WHERE T.TRIP_ID = :NEW.TRIP_ID;
    IF l_trip_available_no_places - :NEW.NO_AVAILABLE_PLACES < 0 THEN
        RAISE_APPLICATION_ERROR(-20002, 'There are not free slots enough');
    END IF;
END;

CREATE OR REPLACE PROCEDURE ModifyNoPlaces(trip_id INT, no_places INT) AS
    l_trip_id INT := trip_id;
BEGIN
    UPDATE TRIP
    SET MAX_NO_PLACES = no_places
    WHERE TRIP_ID = l_trip_id;
END;
```

Oczywiście po wprowadzeniu tej zmiany należy uaktualnić procedury modyfikujące dane. Najlepiej to zrobić tworząc nowe wersje (np. z sufiksem 5)

Uwagi

Należy przygotować raport z wykonania ćwiczenia. Raport powinien zawierać polecenia SQL (między innymi kod widoków, procedur, …), wynik działania oraz krótki komentarz. Raport należy przesłać w formie pliku PDF. W raporcie można posłużyć się zrzutami ekranów. Dodatkowo proszę załączyć kod zaimplementowanych widoków/procedur postaci pliku tekstowego (plik tekstowy z rozszerzeniem sql)
Proszę zwrócić uwagę na formatowanie kodu (struktura)

Punktacja za wykonanie pkt. I ćw (wersja dla SZBD Oracle) - max 2pkt

II. Postgresql PL/pgSQL

Dla chętnych.

Należy wykonać ćwiczenie przy wykorzystaniu SZBD Postgresql. Podobnie jak w pkt I należy przygotować raport i przesłać kod.

Raport należy uzupełnić własnymi wnioskami stanowiącymi porównanie rozwiązań dostępnych w Oracle PL/SQL oraz Postgres PL/pgSQL. Dodatkowo można pokusić się o porównanie tych rozwiązań z językiem T-SQL

Punktacja za wykonanie pkt. II ćw (wersja dla SZBD Postgresql) - max 2pkt