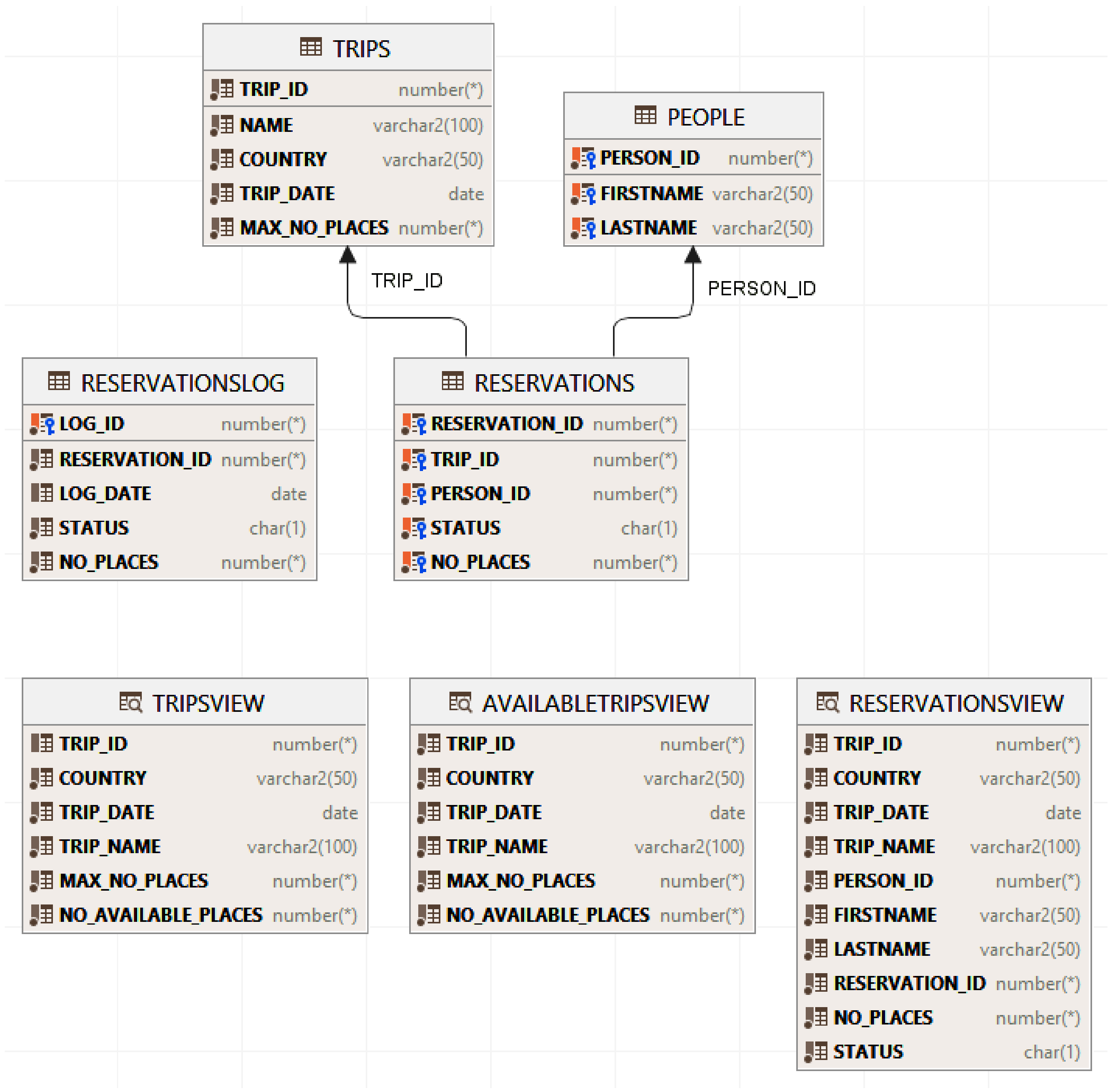
Laboratorium 1

Oracle PL/SQL

Mateusz Łopaciński

Spis treści

1. Schemat bazy danych



2. Tworzenie tabel

2.1. People

```
CREATE TABLE People (
    person_id INT GENERATED ALWAYS AS IDENTITY NOT NULL,
    firstname VARCHAR2(50) NOT NULL,
    lastname VARCHAR2(50) NOT NULL,
    CONSTRAINT People_pk PRIMARY KEY (person_id)
);
```

2.2. Trips

```
CREATE TABLE Trips (
    trip_id INT GENERATED ALWAYS AS IDENTITY NOT NULL,
    name VARCHAR2(100) NOT NULL,
    country VARCHAR2(50) NOT NULL,
    trip_date DATE NOT NULL,
    max_no_places INT NOT NULL,
    CONSTRAINT Trips_pk PRIMARY KEY (trip_id)
);
```

2.3. Reservations

```
CREATE TABLE Reservations (
    reservation_id INT GENERATED ALWAYS AS IDENTITY NOT NULL,
    trip_id INT NOT NULL,
    person_id INT NOT NULL,
    status CHAR(1) NOT NULL,
    no_places INT NOT NULL,
    CONSTRAINT Reservations_pk PRIMARY KEY (reservation_id)
);
```

2.4. ReservationsLog

```
CREATE TABLE ReservationsLog (
    log_id INT GENERATED ALWAYS AS IDENTITY NOT NULL,
    reservation_id INT NOT NULL,
    log_date DATE NOT NULL,
    status CHAR(1) NOT NULL,
    no_places INT NOT NULL,
    CONSTRAINT ReservationsLog_pk PRIMARY KEY (log_id)
);
```

3. Warunki integralnościowe

Poniżej umieściłem warunki integralnościowe, które nie zostały zdefiniowane w kodzie tworzącym tabele.

3.1. Trips

```
ALTER TABLE Trips
ADD CONSTRAINT Trips_chk1 CHECK (max_no_places > 0);
```

3.2. Reservations

```
ALTER TABLE Reservations
ADD CONSTRAINT Reservations_fk1 FOREIGN KEY (person_id)
REFERENCES People(person_id);

ALTER TABLE Reservations
ADD CONSTRAINT Reservations_fk2 FOREIGN KEY (trip_id)
```

```
REFERENCES Trips(trip_id);

ALTER TABLE Reservations
ADD CONSTRAINT Reservations_chk1 CHECK (status IN ('n', 'p', 'c'));

ALTER TABLE Reservations
ADD CONSTRAINT Reservations_chk2 CHECK (no_places > 0);
```

3.3. ReservationsLog

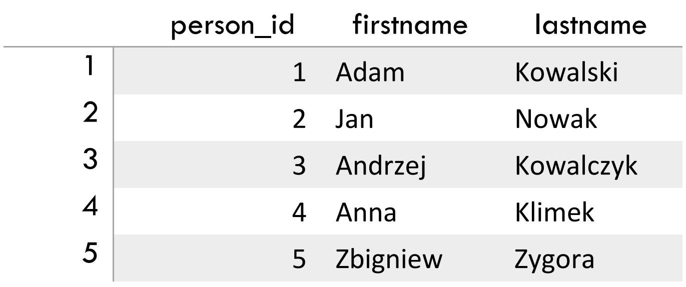
```
ALTER TABLE ReservationsLog
ADD CONSTRAINT ReservationLog_chk1 CHECK (status IN ('n', 'p', 'c'));

ALTER TABLE ReservationsLog
ADD CONSTRAINT ReservationsLog_chk2 CHECK (no_places > 0);
```

4. Wstawianie danych do tabel

4.1. People

```
INSERT INTO People (firstname, lastname)
VALUES ('Adam', 'Kowalski');

INSERT INTO People (firstname, lastname)
VALUES ('Jan', 'Nowak');

INSERT INTO People (firstname, lastname)
VALUES ('Andrzej', 'Kowalczyk');

INSERT INTO People (firstname, lastname)
VALUES ('Anna', 'Klimek');

INSERT INTO People (firstname, lastname)
VALUES ('Zbigniew', 'Zygora');

INSERT INTO People (firstname, lastname)
VALUES ('Rafał', 'Noga');

INSERT INTO People (firstname, lastname)
VALUES ('Aleksandra', 'Sobczak');

INSERT INTO People (firstname, lastname)
VALUES ('Maryla', 'Ordon');

INSERT INTO People (firstname, lastname)
VALUES ('Piotr', 'Słota');

INSERT INTO People (firstname, lastname)
VALUES ('Aleks', 'Stachowiak');

COMMIT;
```

W rezultacie otrzymujemy tabelę:



	person_id	firstname	lastname
1	1	Adam	Kowalski
2	2	Jan	Nowak
3	3	Andrzej	Kowalczyk
4	4	Anna	Klimek
5	5	Zbigniew	Zygora

6	6	Rafał	Noga
7	7	Aleksandra	Sobczak
8	8	Maryla	Ordon
9	9	Piotr	Słota
10	10	Aleks	Stachowiak

Tabela 4.1. People

4.2. Trips

```sql
INSERT INTO Trips (name, country, trip_date, max_no_places)
VALUES ('wycieczka do Paryza', 'Francja', TO_DATE('2021-09-03', 'yyyy-mm-
dd'), 5);

INSERT INTO Trips (name, country, trip_date, max_no_places)
VALUES ('wycieczka do Krakowa', 'Polska', TO_DATE('2022-12-05', 'yyyy-mm-
dd'), 8);

INSERT INTO Trips (name, country, trip_date, max_no_places)
VALUES ('wycieczka do Warszawy', 'Polska', TO_DATE('2022-04-11', 'yyyy-mm-
dd'), 12);

INSERT INTO Trips (name, country, trip_date, max_no_places)
VALUES ('wycieczka do Madrytu', 'Hiszpania', TO_DATE('2022-07-02', 'yyyy-mm-
dd'), 8);

COMMIT;
```

W rezultacie otrzymujemy tabelę:

	trip_id	name	country	trip_date	max_no_places
1	1	wycieczka do Paryza	Francja	2021-09-03	5
2	2	wycieczka do Krakowa	Polska	2022-12-05	8
3	3	wycieczka do Warszawy	Polska	2022-04-11	12
4	4	wycieczka do Madrytu	Hiszpania	2022-07-02	8

Tabela 4.2. Trips

4.3. Reservations

```sql
INSERT INTO Reservations (trip_id, person_id, no_places, status)
VALUES (1, 1, 1, 'n');

INSERT INTO Reservations (trip_id, person_id, no_places, status)
VALUES (1, 2, 2, 'p');

INSERT INTO Reservations (trip_id, person_id, no_places, status)
VALUES (2, 1, 1, 'p');

INSERT INTO Reservations (trip_id, person_id, no_places, status)
VALUES (2, 2, 1, 'c');

INSERT INTO Reservations (trip_id, person_id, no_places, status)
VALUES (2, 4, 2, 'n');

INSERT INTO Reservations (trip_id, person_id, no_places, status)
VALUES (3, 5, 4, 'c');

INSERT INTO Reservations (trip_id, person_id, no_places, status)
VALUES (3, 5, 3, 'n');
```

```
INSERT INTO Reservations (trip_id, person_id, no_places, status)
VALUES (3, 6, 4, 'p');

INSERT INTO Reservations (trip_id, person_id, no_places, status)
VALUES (4, 7, 3, 'p');

INSERT INTO Reservations (trip_id, person_id, no_places, status)
VALUES (4, 9, 1, 'c');

INSERT INTO Reservations (trip_id, person_id, no_places, status)
VALUES (4, 8, 5, 'n');

COMMIT;
```

W rezultacie otrzymujemy tabelę:

	reservation_id	trip_id	person_id	status	no_places
1	1	1	1	n	1
2	2	1	2	p	2
3	3	2	1	p	1
4	4	2	2	c	1
5	5	2	4	n	2
6	6	3	5	c	4
7	7	3	5	n	3
8	8	3	6	p	4
9	9	4	7	p	3
10	10	4	9	c	1
11	11	4	8	n	5

Tabela 4.3. Reservations

5. Widoki

5.1. ReservationsView

reservations(country,trip_date,trip_name, firstname, lastname,reservation_id,no_places,status)

```
CREATE OR REPLACE VIEW ReservationsView
AS
SELECT
    t.trip_id,
    t.country,
    t.trip_date,
    t.name AS trip_name,
    p.person_id,
    p.firstname,
    p.lastname,
    r.reservation_id,
    r.no_places,
    r.status
FROM Trips t
INNER JOIN Reservations r
ON r.trip_id = t.trip_id
INNER JOIN People p
on p.person_id = r.person_id;

SELECT * FROM ReservationsView;
```

W rezultacie otrzymujemy tabelę:

	trip_id	country	trip_date	trip_name	person_id	firstname	lastname	reservation_id	no_places	status
1	1	Francja	2021-09-03	wycieczka do Paryża	1	Adam	Kowalski	1	1	n
2	2	Polska	2022-12-05	wycieczka do Krakowa	1	Adam	Kowalski	3	1	p
3	1	Francja	2021-09-03	wycieczka do Paryża	2	Jan	Nowak	2	2	p
4	2	Polska	2022-12-05	wycieczka do Krakowa	2	Jan	Nowak	4	1	c
5	2	Polska	2022-12-05	wycieczka do Krakowa	4	Anna	Klimek	5	2	n
6	3	Polska	2022-04-11	wycieczka do Warszawy	5	Zbigniew	Zygora	6	4	c
7	3	Polska	2022-04-11	wycieczka do Warszawy	5	Zbigniew	Zygora	7	3	n
8	3	Polska	2022-04-11	wycieczka do Warszawy	6	Rafał	Noga	8	4	p
9	4	Hiszpania	2022-07-02	Wycieczka do Madrytu	7	Aleksandr a	Sobczak	9	3	p
10	4	Hiszpania	2022-07-02	Wycieczka do Madrytu	8	Maryla	Ordon	11	5	n
11	4	Hiszpania	2022-07-02	Wycieczka do Madrytu	9	Piotr	Słota	10	1	c

Tabela 5.1. Tabela utworzona przez widok ReservationsView

5.2. TripsView

trips(country,trip_date, trip_name,max_no_places, no_available_places)

```
CREATE OR REPLACE VIEW TripsView
AS
SELECT
    trip_id,
    country,
    trip_date,
    name AS trip_name,
    max_no_places,
    getAvailablePlaces(trip_id) AS no_available_places
FROM Trips;
```

W rezultacie otrzymujemy tabelę:

	trip_id	country	trip_date	trip_name	max_no_places	no_available_places
1	1	Francja	2021-09-03	wycieczka do Paryża	5	2

2	2	Polska	2022-12-05	wycieczka do Krakowa	8	5
3	3	Polska	2022-04-11	wycieczka do Warszawy	12	5
4	4	Hiszpania	2022-07-02	wycieczka do Madrytu	8	0

Tabela 5.2. Tabela utworzona przez widok TripsView

5.3. AvailableTripsView

trips(country,trip_date, trip_name, max_no_places, no_available_places)

```
CREATE OR REPLACE VIEW AvailableTripsView
AS
SELECT *
FROM TripsView
WHERE trip_date > SYSDATE
    AND no_available_places > 0;
```

W rezultacie otrzymujemy tabelę:

	trip_id	country	trip_date	trip_name	max_no_places	no_available_places
1	2	Polska	2022-12-05	wycieczka do Krakowa	8	5
2	3	Polska	2022-04-11	wycieczka do Warszawy	12	5

Tabela 5.2. Tabela utworzona przez widok AvailableTripsView

6. Obiekty

6.1. TripParticipantObject i TripParticipantsTable

```
CREATE OR REPLACE TYPE TripParticipantObject AS OBJECT (
    firstname VARCHAR2(50),
    lastname VARCHAR2(50),
    reservation_id INT,
    no_places INT,
    status CHAR(1)
);

CREATE OR REPLACE TYPE TripParticipantsTable IS TABLE OF
TripParticipantObject;
```

6.2. PersonReservationObject i PersonReservationsTable

```
CREATE OR REPLACE TYPE PersonReservationObject AS OBJECT (
    trip_id INT,
    country VARCHAR2(50),
    trip_date DATE,
    trip_name VARCHAR2(100),
    reservation_id INT,
    no_places INT,
    status CHAR(1)
);

CREATE OR REPLACE TYPE PeopleReservationsTable IS TABLE OF
PersonReservationObject;
```

6.3. AvailableTripObject i AvailableTripsTable

```
CREATE OR REPLACE TYPE AvailableTripObject AS OBJECT (
    trip_id INT,
    name VARCHAR2(100),
    country VARCHAR2(50),
    trip_date DATE,
    max_no_places INT
);

CREATE OR REPLACE TYPE AvailableTripsTable IS TABLE OF AvailableTripObject;
```

7. Funkcje skalarne

7.1. getBookedPlacesNum

```
CREATE OR REPLACE FUNCTION getBookedPlacesNum(
    p_trip_id Trips.trip_id%TYPE
)
RETURN Reservations.no_places%TYPE
AS
    l_booked_places Reservations.no_places%TYPE;
BEGIN
    SELECT NVL(SUM(no_places), 0)
    INTO l_booked_places
    FROM Reservations
    WHERE trip_id = p_trip_id
        AND status != 'c';

    RETURN l_booked_places;
END;
```

Przykład działania:

```
BEGIN
    DBMS_OUTPUT.PUT_LINE(getBookedPlacesNum(1));
END;
```

Wynik: 3

7.2. getAvailablePlacesNum

```
CREATE OR REPLACE FUNCTION getAvailablePlacesNum(
    p_trip_id Trips.trip_id%TYPE
)
RETURN Reservations.no_places%TYPE
AS
    l_available_places Trips.max_no_places%TYPE;
BEGIN
    SELECT max_no_places - getBookedPlacesNum(trip_id)
    INTO l_available_places
    FROM Trips
    WHERE trip_id = p_trip_id;

    RETURN l_available_places;

    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            RAISE_APPLICATION_ERROR(-20000, 'Trip with id: ' || p_trip_id
|| ' does not exist');
            RETURN NULL;
END;
```

Przykład działania:

```
BEGIN
    DBMS_OUTPUT.PUT_LINE(getAvailablePlacesNum(1));
END;
```

Wynik: 2

7.3. doesTripExist

```
CREATE OR REPLACE FUNCTION doesTripExist(
    p_trip_id Trips.trip_id%TYPE
)
RETURN BOOLEAN
AS
    exist NUMBER;
BEGIN
    SELECT
        CASE
            WHEN EXISTS(SELECT * FROM Trips WHERE trip_id = p_trip_id) THEN 1
            ELSE 0
        END
    INTO exist
    FROM Dual;

    IF exist = 1 THEN
       RETURN TRUE;
    ELSE
       RETURN FALSE;
    END IF;
END;
```

Przykład działania:

```
BEGIN
    IF doesTripExist(1) THEN
        DBMS_OUTPUT.PUT_LINE('Trip exists');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Trip does not exist');
    END IF;
END;
```

Wynik: Trip exists

7.4. doesPersonExist

```
CREATE OR REPLACE FUNCTION doesPersonExist(
    p_person_id People.person_id%TYPE
)
RETURN BOOLEAN
AS
    exist NUMBER;
BEGIN
   SELECT
        CASE
            WHEN EXISTS(SELECT * FROM People WHERE person_id = p_person_id) THEN 1
            ELSE 0
        END
   INTO exist
   FROM Dual;

   IF exist = 1 THEN
```

```
        RETURN TRUE;
    ELSE
        RETURN FALSE;
    END IF;
END;
```

Przykład działania:

```
BEGIN
    IF doesPersonExist(123) THEN
        DBMS_OUTPUT.PUT_LINE('Person exists');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Person does not exist');
    END IF;
END;
```

Wynik: Person does not exists

7.5. doesReservationExist

```
CREATE OR REPLACE FUNCTION doesReservationExist(
    p_reservation_id Reservations.reservation_id%TYPE
)
RETURN BOOLEAN
AS
    exist NUMBER;
BEGIN
   SELECT
        CASE
            WHEN EXISTS(SELECT * FROM Reservations WHERE reservation_id =
p_reservation_id) THEN 1
            ELSE 0
        END
   INTO exist
   FROM Dual;

   IF exist = 1 THEN
       RETURN TRUE;
   ELSE
       RETURN FALSE;
   END IF;
END;
```

Przykład działania:

```
BEGIN
    IF doesReservationExist(10) THEN
        DBMS_OUTPUT.PUT_LINE('Reservation exists');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Reservation does not exist');
    END IF;
END;
```

Wynik: Reservation exists

7.6. hasTripTakenPlace

```
CREATE OR REPLACE FUNCTION hasTripTakenPlace(
    p_trip_id Trips.trip_id%TYPE,
    p_date DATE DEFAULT SYSDATE
)
RETURN BOOLEAN
AS
```

```
    l_trip_date DATE;
BEGIN
    IF NOT doesTripExist(p_trip_id) THEN
        RAISE_APPLICATION_ERROR(-20000, 'There is no trip with id ' ||
p_trip_id || ' in the database');
    END IF;

    SELECT trip_date
    INTO l_trip_date
    FROM Trips
    WHERE trip_id = p_trip_id;

    IF l_trip_date <= p_date THEN
        RETURN TRUE;
    ELSE
        RETURN FALSE;
    END IF;
END;
```

Przykład działania:

```
BEGIN
    IF hasTripTakenPlace(2) THEN
        DBMS_OUTPUT.PUT_LINE('Trip took place before');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Trip has not taken place yet');
    END IF;
END;
```

Wynik: Trip has not taken place yet

8. Funkcje tabelaryczne

8.1. getTripParticipants

```
CREATE OR REPLACE FUNCTION getTripParticipants(
    p_trip_id Trips.trip_id%TYPE
)
RETURN TripParticipantsTable
AS
    l_result TripParticipantsTable;
BEGIN
    IF NOT doesTripExist(p_trip_id) THEN
        RAISE_APPLICATION_ERROR(-20000, 'There are is no trip with id ' ||
p_trip_id || ' in the database');
    END IF;

    SELECT TripParticipantObject(
        firstname,
        lastname,
        reservation_id,
        no_places,
        status
    )
    BULK COLLECT INTO l_result
    FROM ReservationsView
    WHERE trip_id = p_trip_id
        AND status != 'c';

    RETURN l_result;
END;
```

Przykład działania:

```
SELECT * FROM getTripParticipants(3);
```

Rezultat:

	firstname	lastname	reservation_id	no_places	status
1	Zbigniew	Zygora	7	3	n
2	Rafał	Noga	8	4	p

Tabela 8.1. Przykładowa tabela zwrócona przez funkcję getTripParticipants

8.2. getPersonReservations

```
CREATE OR REPLACE FUNCTION getPersonReservations(
    p_person_id People.person_id%TYPE
)
RETURN PeopleReservationsTable
AS
    l_result PeopleReservationsTable;
BEGIN
    IF NOT doesPersonExist(p_person_id) THEN
        RAISE_APPLICATION_ERROR(-20000, 'There is no person with id ' ||
p_person_id || ' in the database');
    END IF;

    SELECT PersonReservationObject(
        trip_id,
        country,
        trip_date,
        trip_name,
        reservation_id,
        no_places,
        status
    )
    BULK COLLECT INTO l_result
    FROM ReservationsView
    WHERE person_id = p_person_id;

    RETURN l_result;
END;
```

Przykład działania:

```
SELECT * FROM getPersonReservations(2);
```

Rezultat:

	trip_id	country	trip_date	trip_name	reservation_id	no_places	status
1	1	Francja	2021-09-03	wycieczka do Paryża	2	2	p
2	2	Polska	2022-12-05	wycieczka do Krakowa	4	1	c

Tabela 8.2. Przykładowa tabela zwrócona przez funkcję getPersonReservations

8.3. getAvailableTripsTo

```
-- (When there are no remaining places, a trip is considered unavailable -
see Spain in examples)
-- (if p_from_date is lower than the current date, a function below will
return only trips which
--  date is between the current date and the p_to_date)
CREATE OR REPLACE FUNCTION getAvailableTripsTo(
    p_country_name Trips.country%TYPE,
    p_from_date DATE,
    p_to_date DATE
)
RETURN AvailableTripsTable
AS
    l_result AvailableTripsTable;
BEGIN
    -- Show warning information if the current date is greater than the
p_from_date
    IF p_from_date < SYSDATE THEN
        DBMS_OUTPUT.PUT_LINE('Warning: Specified trip start date (' ||
p_from_date ||
                             ') is lower than the current date.' || 'Current
date (' || SYSDATE ||
                             ') will be used instead');
    END IF;

    SELECT AvailableTripObject(
        trip_id,
        trip_name,
        country,
        trip_date,
        max_no_places
    )
    BULK COLLECT
    INTO l_result
    FROM AvailableTripsView
    WHERE country = p_country_name
        AND trip_date BETWEEN p_from_date AND p_to_date;

    RETURN l_result;
END;
```

Przykład działania:

```
SELECT * FROM getAvailableTripsTo('Polska', '2020-01-01', '2022-12-31');
```

Rezultat:

	trip_id	name	country	trip_date	max_no_places
1	2	Wycieczka do Krakowa	Polska	2022-12-05	8
2	3	Wycieczka do Warszawy	Polska	2022-04-11	12

Tabela 8.3. Przykładowa tabela zwrócona przez funkcję getAvailableTripsTo

9. Procedury (wersja 1.)

9.1. addReservation

Poniżej znajdują się 1. wersje procedur (zanim zostały dodane triggery, które obsługują sprawdzanie poprawności odpowiednich danych).

```
CREATE OR REPLACE PROCEDURE addReservation(
    p_trip_id Trips.trip_id%TYPE,
    p_person_id People.person_id%TYPE,
    p_no_places Reservations.no_places%TYPE
)
AS
    l_available_places Reservations.no_places%TYPE;
BEGIN
    IF NOT doesPersonExist(p_person_id) THEN
        RAISE_APPLICATION_ERROR(-20001, 'The person with id ' || p_person_id
|| ' does not exit');
    END IF;

    IF hasTripTakenPlace(p_trip_id) THEN
        RAISE_APPLICATION_ERROR(-20001, 'The trip with id ' || p_trip_id || '
took place before');
    END IF;

    IF p_no_places < 1 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Cannot book less than 1 place for a
trip');
    END IF;

    l_available_places := getAvailablePlacesNum(p_trip_id);

    IF l_available_places = 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'There are no available places for a
trip with id ' || p_trip_id);
    ELSIF l_available_places < p_no_places THEN
        RAISE_APPLICATION_ERROR(-20001, 'There are only ' ||
l_available_places ||
                                        ' places available for a trip with id
' || p_trip_id);
    END IF;

    INSERT INTO Reservations (trip_id, person_id, status, no_places)
    VALUES (p_trip_id, p_person_id, 'n', p_no_places);

    COMMIT;
END;
```

Przykład działania:

```
BEGIN
    addReservation(2, 1, 5);
END;
```

Rezerwacje po użyciu procedury:

	reservation_id	trip_id	person_id	status	no_places
1	1	1	1	n	1
2	2	1	2	p	2
3	3	2	1	p	1
4	4	2	2	c	1
5	5	2	4	n	2
6	6	3	5	c	4
7	7	3	5	n	3
8	8	3	6	p	4
9	9	4	7	p	3
10	10	4	9	c	1
11	11	4	8	n	5

12	12	2	1	n	5

Tabela 9.1. Rezerwacje po dodaniu nowej rezerwacji

Jak widzimy, w ostatnim wierszu tabeli 9.1. pojawiła się nowa rezerwacja o zadanych parametrach.

9.2. modifyReservationStatus

```
CREATE OR REPLACE PROCEDURE modifyReservationStatus(
    p_reservation_id Reservations.reservation_id%TYPE,
    p_status Reservations.status%TYPE
)
AS
    l_curr_status Reservations.status%TYPE;
    l_trip_id Reservations.trip_id%TYPE;
    l_no_places Reservations.no_places%TYPE;
    l_available_places Reservations.no_places%TYPE;
BEGIN
    SELECT
        status,
        trip_id,
        no_places
    INTO
        l_curr_status,
        l_trip_id,
        l_no_places
    FROM Reservations
    WHERE reservation_id = p_reservation_id;

    CASE p_status
    WHEN l_curr_status THEN
        DBMS_OUTPUT.PUT_LINE('The reservation with id ' || p_reservation_id
||
                             ' has already the status: ' || p_status);
        RETURN;
    WHEN 'c' THEN
        NULL;
    WHEN 'n' THEN
        RAISE_APPLICATION_ERROR(-20001, 'Cannot change the status of the
reservation with id ' ||
                                          p_reservation_id || ' to: n');
    WHEN 'p' THEN
        -- Check if can make cancelled reservation available (paid) again
        -- (check if there are enough empty places for a trip)
        IF l_curr_status = 'c' THEN
            l_available_places := getAvailablePlacesNum(l_trip_id);

            IF l_available_places < l_no_places THEN
                RAISE_APPLICATION_ERROR(-20001, 'Not enough places available
to update the cancelled reservation status');
            END IF;
        END IF;
    ELSE
        RAISE_APPLICATION_ERROR(-20001, 'Status: ' || p_status || ' is not a
valid reservation status');
    END CASE;

    -- If everything is correct, update the reservation status
    UPDATE Reservations
    SET status = p_status
    WHERE reservation_id = p_reservation_id;

    EXCEPTION
        WHEN NO_DATA_FOUND THEN
```

```
            RAISE_APPLICATION_ERROR(-20001, 'There is no reservation with id
' || p_reservation_id || ' in the database');

    COMMIT;
END;
```

Przykład działania:

```
BEGIN
    modifyReservationStatus(2, 'c');
    modifyReservationStatus(1, 'c');
    modifyReservationStatus(12, 'p');
END;
```

Rezerwacje po użyciu procedury:

	reservation_id	trip_id	person_id	status	no_places
1	1	1	1	c	1
2	2	1	2	c	2
3	3	2	1	p	1
4	4	2	2	c	1
5	5	2	4	n	2
6	6	3	5	c	4
7	7	3	5	n	3
8	8	3	6	p	4
9	9	4	7	p	3
10	10	4	9	c	1
11	11	4	8	n	5
12	12	2	1	p	5

Tabela 9.2. Rezerwacje po modyfikacji statusu wybranych rezerwacji

9.3. modifyReservationNoPlaces

```
CREATE OR REPLACE PROCEDURE modifyReservationNoPlaces(
    p_reservation_id Reservations.reservation_id%TYPE,
    p_no_places Reservations.no_places%TYPE
)
AS
    l_curr_no_places Reservations.no_places%TYPE;
    l_available_places Reservations.no_places%TYPE;
BEGIN
    SELECT
        no_places,
        getAvailablePlacesNum(trip_id)
    INTO
        l_curr_no_places,
        l_available_places
    FROM Reservations
    WHERE reservation_id = p_reservation_id;

    IF p_no_places <= 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'The number of booked places should
be greater than 0');
    ELSIF p_no_places - l_curr_no_places > l_available_places THEN
        RAISE_APPLICATION_ERROR(-20001, 'There are not enough free places.
Max possible number of places to book: ' ||
                                        (l_available_places +
l_curr_no_places));
```

```
    END IF;

    UPDATE Reservations
    SET no_places = p_no_places
    WHERE reservation_id = p_reservation_id;


    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            RAISE_APPLICATION_ERROR(-20001, 'There is no reservation with id
' || p_reservation_id || ' in the database');

    COMMIT;
END;
```

Przykład działania:

```
BEGIN
    modifyReservationNoPlaces(1, 6);
END;
```

Rezerwacje po użyciu procedury:

	reservation_id	trip_id	person_id	status	no_places
1	1	1	1	c	6
2	2	1	2	c	2
3	3	2	1	p	1
4	4	2	2	c	1
5	5	2	4	n	2
6	6	3	5	c	4
7	7	3	5	n	3
8	8	3	6	p	4
9	9	4	7	p	3
10	10	4	9	c	1
11	11	4	8	n	5
12	12	2	1	p	5

Tabela 9.3. Rezerwacje po modyfikacji liczby miejsc w rezerwacji o id równym 1

9.4. modifyMaxNoPlaces

```
CREATE OR REPLACE PROCEDURE modifyMaxNoPlaces(
    p_trip_id Trips.trip_id%TYPE,
    p_max_no_places Trips.max_no_places%TYPE
)
AS
    l_booked_places Reservations.no_places%TYPE;
    l_curr_max_no_places Trips.max_no_places%TYPE;
BEGIN
    SELECT
        max_no_places,
        getBookedPlacesNum(p_trip_id)
    INTO
        l_curr_max_no_places,
        l_booked_places
    FROM Trips
    WHERE trip_id = p_trip_id;

    IF p_max_no_places = l_curr_max_no_places THEN
```

```
        DBMS_OUTPUT.PUT_LINE('The trip with id ' || p_trip_id ||
                             ' has already the maximum number of places set
to ' || l_curr_max_no_places);
        RETURN;
    END IF;

    IF p_max_no_places < l_booked_places THEN
        RAISE_APPLICATION_ERROR(-20001, 'The maximum number of places (' ||
p_max_no_places ||
                                        ') cannot be lower than the total
number of booked places (' || l_booked_places ||
                                        ') for a trip wit id ' || p_trip_id);
    END IF;

    UPDATE Trips
    SET max_no_places = p_max_no_places
    WHERE trip_id = p_trip_id;

    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            RAISE_APPLICATION_ERROR(-20001, 'There is no trip with id ' ||
p_trip_id || ' in the database');
    COMMIT;
END;
```

Przykład działania:

```
BEGIN
    modifyMaxNoPlaces(4, 8);
    modifyMaxNoPlaces(3, 7);
    modifyMaxNoPlaces(2, 10);
END;
```

Wycieczki po użyciu procedury:

	trip_id	name	country	trip_date	max_no_places
1	1	wycieczka do Paryza	Francja	2021-09-03	5
2	2	wycieczka do Krakowa	Polska	2022-12-05	10
3	3	wycieczka do Warszawy	Polska	2022-04-11	7
4	4	wycieczka do Madrytu	Hiszpania	2022-07-02	8

Tabela 9.4. Wycieczki po modyfikacji maksymalnej liczby miejsc

10. Triggery

10.1. AI_ReservationInsert

```
CREATE OR REPLACE TRIGGER AI_ReservationInsert
AFTER INSERT
ON Reservations
FOR EACH ROW
BEGIN
    INSERT INTO ReservationsLog (reservation_id, log_date, status,
no_places)
    VALUES (:NEW.reservation_id, SYSDATE, :NEW.status, :NEW.no_places);
END;
```

Przykład działania:

```
BEGIN
    addReservation(2, 1, 1);
END;
```

Tabela ReservationsLog po dodaniu rezerwacji do tabeli Reservations:

	log_id	reservation_id	log_date	status	no_places
1	1	13	2022-03-12 02:05:23	n	1

Tabela 10.1. Tabela ReservationsLog po dodaniu nowej rezerwacji

10.2. AU_ReservationStatusUpdate

```
CREATE OR REPLACE TRIGGER AU_ReservationStatusUpdate
AFTER UPDATE
OF status ON Reservations
FOR EACH ROW
BEGIN
    IF :NEW.status != :OLD.status THEN
        INSERT INTO ReservationsLog (reservation_id, log_date, status,
no_places)
        VALUES (:NEW.reservation_id, SYSDATE, :NEW.status, :NEW.no_places);
    END IF;
END;
```

Przykład działania:

```
DECLARE
    l_reservation_id Reservations.reservation_id%TYPE;
BEGIN
    SELECT MAX(reservation_id)
    INTO l_reservation_id
    FROM Reservations;

    modifyReservationStatus(l_reservation_id, 'p');
END;
```

Tabela ReservationsLog po zaktualizowaniu statusu rezerwacji:

	log_id	reservation_id	log_date	status	no_places
1	1	13	2022-03-12 02:05:23	n	1
2	2	13	2022-03-12 02:10:14	p	1

Tabela 10.2. Tabela ReservationsLog po zaktualizowaniu statusu rezerwacji

10.3. AU_ReservationNoPlacesUpdate

```
CREATE OR REPLACE TRIGGER AU_ReservationNoPlacesUpdate
AFTER UPDATE
OF no_places ON Reservations
FOR EACH ROW
BEGIN
    IF :NEW.no_places != :OLD.no_places THEN
        INSERT INTO ReservationsLog (reservation_id, log_date, status,
no_places)
        VALUES (:NEW.reservation_id, SYSDATE, :NEW.status, :NEW.no_places);
    END IF;
END;
```

Przykład działania:

```
DECLARE
    l_reservation_id Reservations.reservation_id%TYPE;
BEGIN
    SELECT MAX(reservation_id)
    INTO l_reservation_id
    FROM Reservations;

    modifyReservationNoPlaces(l_reservation_id, 2);
```

```
END;
```

Tabela ReservationsLog po zaktualizowaniu liczby zarezerwowanych miejsc:

	log_id	reservation_id	log_date	status	no_places
1	1	13	2022-03-12 02:05:23	n	1
2	2	13	2022-03-12 02:10:14	p	1
3	3	13	2022-03-12 02:13:16	p	2

Tabela 10.2. Tabela ReservationsLog po zaktualizowaniu liczby zarezerwowanych miejsc

10.4. BI_ ReservationInsert

```
CREATE OR REPLACE TRIGGER BI_ReservationInsert
BEFORE INSERT
ON Reservations
FOR EACH ROW
DECLARE
    l_available_places Reservations.no_places%TYPE;
BEGIN
    IF NOT doesPersonExist(:NEW.person_id) THEN
        RAISE_APPLICATION_ERROR(-20001, 'The person with id ' ||
:NEW.person_id || ' does not exit');
    END IF;

    IF :NEW.no_places < 1 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Cannot book less than 1 place for a
trip');
    END IF;

    l_available_places := getAvailablePlacesNum(:NEW.trip_id);

    IF l_available_places = 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'There are no available places for a
trip with id ' || :NEW.trip_id);
    ELSIF l_available_places < :NEW.no_places THEN
        RAISE_APPLICATION_ERROR(-20001, 'There are only ' ||
l_available_places ||
                                        ' places available for a trip with id
' || :NEW.trip_id);
    END IF;
END;
```

Ten trigger pozwala na wyodrębnienie części funkcjonalności z procedury **addReservation**. Ponieważ nie wprowadza on nowej funkcjonalności, a jedynie pozwala na uproszczenie kodu procedury (nowy kod procedury zamieściłem w kolejnej sekcji), nie zamieszczam tutaj osobnych przykładów działania (działanie triggera wraz ze zmodyfikowaną procedurą jest analogiczne do działania wcześniejszej implementacji procedury **addReservation**).

10.5. BU_ ReservationStatusUpdate

```
CREATE OR REPLACE TRIGGER BU_ReservationStatusUpdate
BEFORE UPDATE
OF status ON Reservations
FOR EACH ROW
DECLARE
    PRAGMA AUTONOMOUS_TRANSACTION;
    l_available_places Reservations.no_places%TYPE;
BEGIN
    CASE :NEW.status
    WHEN :OLD.status THEN
        DBMS_OUTPUT.PUT_LINE('The reservation with id ' ||
:NEW.reservation_id ||
                             ' has already the status: ' || :NEW.status);
        RETURN;
    WHEN 'c' THEN
        NULL;
    WHEN 'n' THEN
        RAISE_APPLICATION_ERROR(-20001, 'Cannot change the status of the
reservation with id ' ||
                                :NEW.reservation_id || ' to: n');
    WHEN 'p' THEN
        -- Check if can make cancelled reservation available (paid) again
        -- (check if there are enough empty places for a trip)
        IF :OLD.status = 'c' THEN
            l_available_places := getAvailablePlacesNum(:NEW.trip_id);

            IF l_available_places < :NEW.no_places THEN
                RAISE_APPLICATION_ERROR(-20001, 'Not enough places available
to update the cancelled reservation status');
            END IF;
        END IF;
    ELSE
        RAISE_APPLICATION_ERROR(-20001, 'Status: ' || :NEW.status || ' is not
a valid reservation status');
    END CASE;
END;
```

Ponownie, z tego samego powodu, co powyżej, nie zamieszczam przykładów.

10.6. BU_ ReservationNoPlacesUpdate

```
CREATE OR REPLACE TRIGGER BU_ReservationNoPlacesUpdate
BEFORE UPDATE
OF no_places ON Reservations
FOR EACH ROW
DECLARE
    PRAGMA AUTONOMOUS_TRANSACTION;
    l_available_places Reservations.no_places%TYPE;
BEGIN
    l_available_places := getAvailablePlacesNum(:NEW.trip_id);

    IF :NEW.no_places <= 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'The number of booked places should
be greater than 0');
    ELSIF :NEW.no_places - :OLD.no_places > l_available_places THEN
        RAISE_APPLICATION_ERROR(-20001, 'There are not enough free places.
Max possible number of places to book: ' ||
                                       (l_available_places +
:OLD.no_places));
    END IF;
END;
```

Ponownie, z tego samego powodu, co powyżej, nie zamieszczam przykładów.

10.7. BU_TripMaxNoPlacesUpdate

```
CREATE OR REPLACE TRIGGER BU_TripMaxNoPlacesUpdate
BEFORE UPDATE
OF max_no_places ON Trips
FOR EACH ROW
DECLARE
    l_booked_places Reservations.no_places%TYPE;
BEGIN
    l_booked_places := getBookedPlacesNum(:NEW.trip_id);

    IF :NEW.max_no_places = :OLD.max_no_places THEN
        DBMS_OUTPUT.PUT_LINE('The trip with id ' || :NEW.trip_id ||
                             ' has already the maximum number of places set
to ' || :OLD.max_no_places);
        RETURN;
    END IF;

    IF :NEW.max_no_places < l_booked_places THEN
        RAISE_APPLICATION_ERROR(-20001, 'The maximum number of places (' ||
:NEW.max_no_places ||
                                        ') cannot be lower than the total
number of booked places (' || l_booked_places ||
                                        ') for a trip wit id ' ||
:NEW.trip_id);
    END IF;
END;
```

Ponownie, z tego samego powodu, co powyżej, nie zamieszczam przykładów.

11. Procedury (wersja 2.)

Poniżej znajdują się 2. wersje procedur (z procedur została usunięta zawartość przeniesiona do triggerów).

11.1. addReservation

```
CREATE OR REPLACE PROCEDURE addReservation(
    p_trip_id Trips.trip_id%TYPE,
    p_person_id People.person_id%TYPE,
    p_no_places Reservations.no_places%TYPE
)
AS
BEGIN
    IF hasTripTakenPlace(p_trip_id) THEN
        RAISE_APPLICATION_ERROR(-20001, 'The trip with id ' || p_trip_id || ' took
place before');
    END IF;

    INSERT INTO Reservations (trip_id, person_id, status, no_places)
    VALUES (p_trip_id, p_person_id, 'n', p_no_places);

    COMMIT;
END;
```

11.2. modifyReservationStatus

```
CREATE OR REPLACE PROCEDURE modifyReservationStatus(
    p_reservation_id Reservations.reservation_id%TYPE,
    p_status Reservations.status%TYPE
)
AS
BEGIN
    IF NOT doesReservationExist(p_reservation_id) THEN
        RAISE_APPLICATION_ERROR(-20001, 'There is no reservation with id ' ||
p_reservation_id || ' in the database');
    END IF;

    -- If everything is correct, update the reservation status
    UPDATE Reservations
    SET status = p_status
    WHERE reservation_id = p_reservation_id;

    COMMIT;
END;
```

```
CREATE OR REPLACE PROCEDURE modifyReservationNoPlaces(
    p_reservation_id Reservations.reservation_id%TYPE,
    p_no_places Reservations.no_places%TYPE
)
AS
BEGIN
    IF NOT doesReservationExist(p_reservation_id) THEN
        RAISE_APPLICATION_ERROR(-20001, 'There is no reservation with id ' ||
p_reservation_id || ' in the database');
    END IF;

    UPDATE Reservations
    SET no_places = p_no_places
    WHERE reservation_id = p_reservation_id;

    COMMIT;
END;
```

11.3. modifyMaxNoPlaces

```
CREATE OR REPLACE PROCEDURE modifyMaxNoPlaces(
    p_trip_id Trips.trip_id%TYPE,
    p_max_no_places Trips.max_no_places%TYPE
)
AS
BEGIN
    IF NOT doesTripExist(p_trip_id) THEN
        RAISE_APPLICATION_ERROR(-20001, 'There is no trip with id ' || p_trip_id
|| ' in the database');
    END IF;

    UPDATE Trips
    SET max_no_places = p_max_no_places
    WHERE trip_id = p_trip_id;
END;
```