Julia Smerdel

**Raport 1**

Spis treści

[Schemat bazy danych 4](#_Toc130235581)

[Tabele 6](#_Toc130235582)

[Tabela Person 7](#_Toc130235583)

[Tabela Trip 8](#_Toc130235584)

[Tabela Reservation 9](#_Toc130235585)

[Tabela Country 10](#_Toc130235586)

[Tabela Log 11](#_Toc130235587)

[Warunki integralnościowe 12](#_Toc130235588)

[Trips 13](#_Toc130235589)

[Log 13](#_Toc130235590)

[Reservation 13](#_Toc130235591)

[Wstawianie danych do bazy 14](#_Toc130235592)

[Widoki 17](#_Toc130235593)

[Widok Reservations 18](#_Toc130235594)

[Widok Trips 19](#_Toc130235595)

[Widok AvailableTrip 20](#_Toc130235596)

[Procedury 21](#_Toc130235597)

[Procedura AddReservation 22](#_Toc130235598)

[Procedura ModifyReservationStatus 24](#_Toc130235599)

[Procedura ModifyNoPlaces 27](#_Toc130235600)

[Procedura countAvailablePlaces 28](#_Toc130235601)

[Funkcje 29](#_Toc130235602)

[Funkcje skalarne 30](#_Toc130235603)

[Funkcja getBookedPlaces 30](#_Toc130235604)

[Funkcja getLeftPlaces 30](#_Toc130235605)

[Funkcja tripExistence 31](#_Toc130235606)

[Funkcja personExistence 32](#_Toc130235607)

[Funkcja reservationExistence 32](#_Toc130235608)

[Funkcja isDateCorrect 33](#_Toc130235609)

[Funkcje zwracające tabele 34](#_Toc130235610)

[Funkcja TripParticipants 34](#_Toc130235611)

[Funkcja PersonReservations 35](#_Toc130235612)

[Funkcja AvailableTripsTo 36](#_Toc130235613)

[Triggery 38](#_Toc130235614)

[Trigger AddReservationTrigger 39](#_Toc130235615)

[Trigger BeforeAddingReservationTrigger 41](#_Toc130235616)

[Trigger BeforeChangingReservationStatus 42](#_Toc130235617)

[Trigger BeforeChangingMaxPlacesTrigger 44](#_Toc130235618)

[Obiekty 45](#_Toc130235619)

Stworzona baza danych dostępna jest pod nazwą **BD\_409614**

# Schemat bazy danych

Obraz zawierający tekst

Opis wygenerowany automatycznie

# Tabele

## Tabela Person

*Reprezentacja osób*

**Klucz główny:** person\_id

**Imię:** firstname

**Nazwisko**: lastname

create table person  
(  
 person\_id int generated always as identity not null,  
 firstname varchar(50),  
 lastname varchar(50),  
 constraint person\_pk primary key ( person\_id ) enable  
);

## Tabela Trip

*Reprezentacja wycieczek*

**Klucz główny:** trip\_id

**Nazwa wycieczki:** trip\_name

**ID kraju wycieczki:** country\_id

**Data wycieczki**: trip\_date

**Maksymalna liczba miejsc**: max\_no\_places

create table trip  
(  
 trip\_id int generated always as identity not null,  
 trip\_name varchar(100),  
 country\_id int,  
 trip\_date date,  
 max\_no\_places int,  
 constraint trip\_pk primary key ( trip\_id ) enable  
);

Później dodano również redundantne pole **no\_available\_places** reprezentujące liczbę dostępnych miejsc.

create table trip  
(  
 trip\_id int generated always as identity not null,  
 trip\_name varchar(100),  
 country\_id int,  
 trip\_date date,  
 max\_no\_places int,  
 no\_available\_places int,  
 constraint trip\_pk primary key ( trip\_id ) enable  
);

## Tabela Reservation

*Reprezentacja rezerwacji*

**Klucz główny:** reservation\_id

**ID wycieczki:** trip\_id

**ID osoby:** person\_id

**Status:** status

create table reservation  
(  
 reservation\_id int generated always as identity not null,  
 trip\_id int,  
 person\_id int,  
 status char(1),  
  
 constraint reservation\_pk primary key ( reservation\_id ) enable  
);

## Tabela Country

*Reprezentacja krajów*

**Klucz główny:** country\_id

**Nazwa kraju:** country\_name

create table country(  
 country\_id int generated always as identity not null,  
 country\_name varchar(50),  
 constraint country\_pk primary key (country\_id) enable  
);

## Tabela Log

*Reprezentacja zmian*

**Klucz główny:** log\_id

**ID rezerwacji:** reservation\_id

**Data zmiany:** log\_date

**Status:** status

create table log  
(  
 log\_id int generated always as identity not null,  
 reservation\_id int not null,  
 log\_date date not null,  
 status char(1),  
 constraint log\_pk primary key ( log\_id ) enable  
);

# Warunki integralnościowe

## Trips

alter table trip  
add constraint trip\_fk foreign key  
(country\_id) references country(country\_id) enable;

## Log

alter table log  
add constraint log\_chk1 check  
(status in ('N','P','C')) enable;  
  
alter table log  
add constraint log\_fk1 foreign key  
( reservation\_id ) references reservation ( reservation\_id ) enable;

## Reservation

alter table reservation  
add constraint reservation\_fk1 foreign key  
( person\_id ) references person ( person\_id ) enable;  
  
alter table reservation  
add constraint reservation\_fk2 foreign key  
( trip\_id ) references trip ( trip\_id ) enable;  
  
alter table reservation  
add constraint reservation\_chk1 check  
(status in ('N','P','C')) enable;

# Wstawianie danych do bazy

--country  
insert into country(country\_name)  
values ('Polska');  
  
insert into country(country\_name)  
values ('Francja');  
  
insert into country(country\_name)  
values ('Czechy');

-- trip  
  
insert into trip(trip\_name, country\_id, trip\_date, max\_no\_places, no\_available\_places)  
values ('Ahoj przygodo', 1, to\_date('2022-06-10','YYYY-MM-DD'), 10, 8);  
  
insert into trip(trip\_name, country\_id, trip\_date, max\_no\_places, no\_available\_places)  
values (N'Łoscypek', 1, to\_date('2023-07-03','YYYY-MM-DD'), 4, 4);  
  
insert into trip(trip\_name, country\_id, trip\_date, max\_no\_places, no\_available\_places)  
values (N'Żelipapą', 2, to\_date('2023-05-01','YYYY-MM-DD'), 5, 4);  
  
insert into trip(trip\_name, country\_id, trip\_date, max\_no\_places, no\_available\_places)  
values ('Krecik puka w taborecik', 3, to\_date('2023-05-01','YYYY-MM-DD'), 2, 2);

-- person  
  
insert into person(firstname, lastname)  
values (N'Paweł', N'Gaweł');  
  
insert into person(firstname, lastname)  
values ('Joanna', N'Bułeczka');  
  
insert into person(firstname, lastname)  
values ('Katarzyna', 'Pierzyna');  
  
insert into person(firstname, lastname)  
values ('Adam', 'Padam');  
  
insert into person(firstname, lastname)  
values ('Jakub', N'Burak');  
  
insert into person(firstname, lastname)  
values ('Piotr', N'Łotr');  
  
insert into person(firstname, lastname)  
values ('Anna', 'Wanna');

insert into person(firstname, lastname)  
values ('Helena', 'Polena');  
  
insert into person(firstname, lastname)  
values ('Dominik', N'Chałka');  
  
insert into person(firstname, lastname)  
values ('Szymon', 'Dywan');

-- reservation  
-- trip 1  
insert into reservation(trip\_id, person\_id, status)  
values (1, 1, 'P');  
  
insert into reservation(trip\_id, person\_id, status)  
values (1, 2, 'N');  
  
-- trip 2  
insert into reservation(trip\_id, person\_id, status)  
values (2, 1, 'P');  
  
insert into reservation(trip\_id, person\_id, status)  
values (2, 4, 'C');  
  
-- trip 3  
insert into reservation(trip\_id, person\_id, status)  
values (3, 9, 'P');

# Widoki

## Widok Reservations

*Wyświetla wszystkie rezerwacje*

create view Reservations  
as  
 select c.country\_name, t.trip\_date, t.trip\_name, t.trip\_id,  
 p.firstname, p.lastname, p.person\_id, r.reservation\_id,  
 r.status, GETLEFTPLACES(t.trip\_id) as AvailablePlaces  
  
 from reservation r inner join trip t on r.trip\_id = t.trip\_id  
 inner join country c on t.country\_id = c.country\_id  
 inner join person p on r.person\_id = p.person\_id;  
commit;

Zmiana po dodaniu redundantnego pola w tabeli trips

create view ReservationsViewChanged  
as  
 select c.country\_name, t.trip\_date, t.trip\_name, t.trip\_id,  
 p.firstname, p.lastname, p.person\_id, r.reservation\_id,  
 r.status, t.no\_available\_places as AvailablePlaces  
  
 from reservation r inner join trip t on r.trip\_id = t.trip\_id  
 inner join country c on t.country\_id = c.country\_id  
 inner join person p on r.person\_id = p.person\_id;  
commit;

## Widok Trips

*Wyświetla wycieczki*

create view Trips  
as  
 select c.country\_name, t.trip\_date as tripDate, t.trip\_name, t.trip\_id,  
 getBookedPlaces(t.trip\_id) as bookedPlaces,  
 getLeftPlaces(t.trip\_id) as availablePlaces  
 from country c inner join trip t on c.country\_id = t.country\_id;  
commit;

Zmiana po dodaniu redundantnego pola w tabeli trips

create view TripsViewChanged  
as  
 select c.country\_name, t.trip\_date as tripDate, t.trip\_name, t.trip\_id,  
 t.max\_no\_places - t.no\_available\_places as bookedPlaces,  
 t.no\_available\_places as availablePlaces  
 from country c inner join trip t on c.country\_id = t.country\_id;  
commit;

## Widok AvailableTrip

*Wyświetla dostępne wycieczki*

create view AvailableTrips  
as  
 select \* from Trips  
 where availablePlaces > 0 and tripDate > SYSDATE;  
commit;

Zmiana po dodaniu redundantnego pola w tabeli trips

create view AvailableTripsViewChanged  
as  
 select \* from TripsViewChanged  
 where availablePlaces > 0 and tripDate > SYSDATE;  
commit;

# Procedury

## Procedura AddReservation

*Dodaje rezerwacje*

create or replace procedure AddReservation(tripID int, personID int)  
as  
 availablePlaces int;  
begin  
 if not personExistence(personID) then  
 raise\_application\_error(-20001, 'Person does not exist');  
 end if;  
 if not tripExistence(tripID) then  
 raise\_application\_error(-20001, 'Trip does not exist');  
 end if;  
 if not isTheDateCorrect(tripID) then  
 raise\_application\_error(-20001, 'The trip has already taken place');  
 end if;  
  
  
 availablePlaces := getLeftPlaces(tripID);  
  
 if availablePlaces = 0 then  
 raise\_application\_error(-20001, 'There are no spots for this trip');  
 end if;  
  
  
  
 insert into Reservation(trip\_id, person\_id, status)  
 values (tripID, personID, 'N');  
end;  
commit;

**Modyfikacja – użycie triggerów**

create or replace procedure AddReservationAfterTriggerMod(tripID int, personID int)  
as  
begin  
 insert into Reservation(trip\_id, person\_id, status)  
 values (tripID, personID, 'N');  
end;  
commit;

**Modyfikacja – dodanie pola no\_available\_places**

create or replace procedure AddReservationChanged(tripID int, personID int)  
as  
begin  
  
 insert into Reservation(trip\_id, person\_id, status)  
 values (tripID, personID, 'N');  
  
-- update Trip set no\_available\_places = getAvailablePlaces(tripID)- 1  
-- where trip\_id = tripID;  
end;  
commit;

## Procedura ModifyReservationStatus

*Modyfikuje status rezerwacji*

create or replace procedure ModifyReservationStatus(reservationID int, newStatus char)  
as  
 availablePlaces int;  
 tripID int;  
 currStatus char;  
begin  
  
 select trip\_id, status into tripID, currStatus from Reservation  
 where reservationID=reservation\_id;  
  
 availablePlaces := getLeftPlaces(tripID);  
  
 if not reservationExistence(reservationID) then  
 raise\_application\_error(-20001, 'Reservation does not exist');  
 end if;  
 if not tripExistence(tripID) then  
 raise\_application\_error(-20001, 'Trip does not exist');  
 end if;  
  
  
 case newStatus  
 when currStatus then  
 raise\_application\_error(-20001,'Reservation already has this status');  
  
 when 'N' then  
 raise\_application\_error(-20001, 'Cannot change status to NEW');  
  
 when 'C' then  
 null;  
  
 when 'P' then  
 if currStatus = 'C' then  
 if availablePlaces = 0 then  
 raise\_application\_error(-20001, 'Not enough places available');  
 end if;  
 end if;  
  
 else  
 raise\_application\_error(-20001, 'Wrong status given');  
 end case;  
  
  
 update Reservation set status=newStatus where reservation\_id=reservationID;  
  
end;  
commit;

**Modyfikacja – dodanie tabeli log**

create or replace procedure ModifyReservationStatusMod6(reservationID int, newStatus char)  
as  
 availablePlaces int;  
 tripID int;  
 currStatus char;  
begin  
  
 select trip\_id, status into tripID, currStatus from Reservation  
 where reservationID=reservation\_id;  
  
 availablePlaces := getLeftPlaces(tripID);  
  
 if not reservationExistence(reservationID) then  
 raise\_application\_error(-20001, 'Reservation does not exist');  
 end if;  
 if not tripExistence(tripID) then  
 raise\_application\_error(-20001, 'Trip does not exist');  
 end if;  
  
  
 case newStatus  
 when currStatus then  
 raise\_application\_error(-20001,'Reservation already has this status');  
  
 when 'N' then  
 raise\_application\_error(-20001, 'Cannot change status to NEW');  
  
 when 'C' then  
 null;  
  
 when 'P' then  
 if currStatus = 'C' then  
 if availablePlaces = 0 then  
 raise\_application\_error(-20001, 'Not enough places available');  
 end if;  
 end if;  
  
 else  
 raise\_application\_error(-20001, 'Wrong status given');  
 end case;  
  
  
 update Reservation set status=newStatus where reservation\_id=reservationID;  
  
 insert into Log(reservation\_id, log\_date, status) values  
 (reservationID, sysdate, newStatus);  
  
end;  
commit;

**Modyfikacja – użycie triggerów**

create or replace procedure ModifyReservationStatusAfterTriggerMod(reservationID int, newStatus char)  
as  
begin  
  
 if not reservationExistence(reservationID) then  
 raise\_application\_error(-20001, 'Reservation does not exist');  
 end if;  
  
 update Reservation set status=newStatus where reservation\_id=reservationID;  
  
end;  
commit;

**Modyfikacja – dodanie pola no\_available\_places**

create or replace procedure ModifyNoPlacesChanged(tripID int, noPlaces int)  
as  
begin  
 update Trip set max\_no\_places = noPlaces where tripID = trip\_id;  
 begin  
 countAvailablePlaces(tripID);  
 end;  
end;  
  
commit;

## Procedura ModifyNoPlaces

*Modyfikuje maksymalną liczbę miejsc wycieczki*

create or replace procedure ModifyNoPlaces(tripID int, noPlaces int)  
as  
 currPlaces int;  
 availablePlaces int;  
begin  
 select max\_no\_places into currPlaces from Trip where tripID = trip\_id;  
 availablePlaces := getLeftPlaces(tripID);  
  
 if not tripExistence(tripID) then  
 raise\_application\_error(-20001, 'Trip with this id does not exist');  
 end if;  
  
 if currPlaces - availablePlaces > noPlaces then  
 raise\_application\_error(-20001, 'The amount of booked places is greater than new max\_no\_places value.');  
 end if;  
  
  
 update Trip set max\_no\_places = noPlaces where tripID = trip\_id;  
end;  
commit;

**Modyfikacja – dodanie pola no\_available\_places**

create or replace procedure ModifyNoPlacesChanged(tripID int, noPlaces int)  
as  
begin  
 update Trip set max\_no\_places = noPlaces where tripID = trip\_id;  
 begin  
 countAvailablePlaces(tripID);  
 end;  
end;

## Procedura countAvailablePlaces

*Zmienia dostępną liczbę miejsc (użyto podczas dodania redundantnego pola no\_available\_places do tabeli Trip)*

create or replace procedure countAvailablePlaces(tripID int)  
as  
 result int;  
begin  
 select max\_no\_places - getBookedPlaces(tripID) into result from Trip where tripID = trip\_id;  
  
 update Trip set no\_available\_places = result where trip\_id=tripID;  
  
end;  
commit;

# Funkcje

## Funkcje skalarne

## Funkcja getBookedPlaces

*Funkcja do obliczania liczby zabookowanych miejsc*

create or replace function getBookedPlaces(tripID int)  
 return int  
as  
 result int;  
begin  
 select *count*(r.status) into result  
 from reservation r inner join trip t on r.trip\_id = t.trip\_id  
 where r.trip\_id = tripID and r.status != 'C';  
  
 return result;  
end;  
commit;

## Funkcja getLeftPlaces

*Funkcja do obliczania liczby dostępnych miejsc*

create or replace function getLeftPlaces(tripID int)  
 return int  
as  
 result int;  
begin  
 select t.max\_no\_places - getBookedPlaces(tripID) into result  
 from trip t where tripID = t.trip\_id;  
  
 return result;  
end;  
commit;

***Modyfikacja – dodanie pola no\_available\_places***

create or replace function getAvailablePlaces(tripID int)  
 return int  
as  
 result int;  
begin  
 select no\_available\_places into result from Trip where tripID=trip\_id;  
  
 return result;  
end;  
commit;

## Funkcja tripExistence

*Funkcja sprawdza, czy wycieczka istnieje*

create or replace function tripExistence(tripID int)  
 return boolean  
as  
 exist number;  
begin  
 select case  
 when *exists*(select \* from trip where trip.trip\_id = tripID) then 1  
 else 0  
 end  
 into exist from dual;  
  
 if exist = 1 then  
 return true;  
 else  
 return false;  
 end if;  
end;  
commit;

## Funkcja personExistence

*Funkcja sprawdza, czy osoba istnieje*

create or replace function personExistence(personID int)  
 return boolean  
as  
 exist number;  
begin  
 select case  
 when *exists*(select \* from person where person.person\_id = personID) then 1  
 else 0  
 end  
 into exist from dual;  
  
 if exist = 1 then  
 return true;  
 else  
 return false;  
 end if;  
end;  
commit;

## Funkcja reservationExistence

*Funkcja sprawdza, czy rezerwacja istnieje*

create or replace function reservationExistence(reservationID int)  
 return boolean  
as  
 exist number;  
begin  
 select case  
 when *exists*(select \* from reservation where reservation.reservation\_id = reservationID) then 1  
 else 0  
 end  
 into exist from dual;  
  
 if exist = 1 then  
 return true;  
 else  
 return false;  
 end if;  
end;  
commit;

## Funkcja isDateCorrect

*Funkcja sprawdza czy data wycieczki jest poprawna – czy już się odbyła*

-- wyświetla klientów, którzy zamówili więcej niż x zamówień za minimalną kwotę y  
create or replace function isTheDateCorrect(tripID int, givenDate date default SYSDATE)  
 return boolean  
as  
 result date;  
begin  
 if not tripExistence(tripID) then  
 raise\_application\_error(-2000, 'The trip does not exist');  
 end if;  
  
 select trip\_date into result from trip where trip\_id = tripID;  
  
 if givenDate >= result then --false when the trip has taken place earlier  
 return false;  
 else  
 return true;  
 end if;  
end;  
commit;

## Funkcje zwracające tabele

## Funkcja TripParticipants

*Funkcja pokazuje uczestników wycieczki*

create or replace function TripParticipants(tripID int)  
 return TripParticipantsTable  
as  
 result TripParticipantsTable;  
begin  
 if not tripExistence(tripID) then  
 raise\_application\_error(-2000, 'There are no such trip in the database');  
 end if;  
  
 select TripParticipant(firstname, lastname, person\_id, trip\_name, trip\_date)  
 bulk collect into result  
 from Reservations  
 where tripID = trip\_id;  
  
 return result;  
end;  
commit;

**Modyfikacja – dodanie pola no\_available\_places**

create or replace function TripParticipantsChanged(tripID int)  
 return TripParticipantsTable  
as  
 result TripParticipantsTable;  
begin  
 if not tripExistence(tripID) then  
 raise\_application\_error(-2000, 'There are no such trip in the database');  
 end if;  
  
 select TripParticipant(firstname, lastname, person\_id, trip\_name, trip\_date)  
 bulk collect into result  
 from ReservationsViewChanged  
 where tripID = trip\_id;  
  
 return result;  
end;  
commit;

## Funkcja PersonReservations

*Funkcja wyświetla rezerwacje danej osoby*

create or replace function PersonReservations(personID int)  
 return PersonReservationsTable  
as  
 result PersonReservationsTable;  
begin  
 if not personExistence(personID) then  
 raise\_application\_error(-2000, 'There are no such person in the database');  
 end if;  
  
 select PersonReservationsO(trip\_name, trip\_date, trip\_id, country\_name, reservation\_id, status)  
 bulk collect into result  
 from Reservations  
 where personID=person\_id;  
  
 return result;  
end;  
commit;

**Modyfikacja – dodanie pola no\_available\_places**

create or replace function PersonReservationsChanged(personID int)  
 return PersonReservationsTable  
as  
 result PersonReservationsTable;  
begin  
 if not personExistence(personID) then  
 raise\_application\_error(-2000, 'There are no such person in the database');  
 end if;  
  
 select PersonReservationsO(trip\_name, trip\_date, trip\_id, country\_name, reservation\_id, status)  
 bulk collect into result  
 from ReservationsViewChanged  
 where personID=person\_id;  
  
 return result;  
end;  
commit;

## Funkcja AvailableTripsTo

*Funkcja wyświetla dostępne wycieczki (w podanym terminie do podanego kraju)*

create or replace function AvailableTripsTo(countryName varchar, dateFrom date, dateTo date)  
 return AvailableTripsTable  
as  
 result AvailableTripsTable;  
begin  
 if dateFrom > dateTo then raise\_application\_error(-2000, 'Wrong date given');  
 end if;  
  
 if dateFrom < sysdate then raise\_application\_error(-2000, 'Wrong date given');  
 end if;  
  
 select AvailableTripsO(trip\_id, trip\_name, country\_name, tripDate, availablePlaces)  
 bulk collect into result  
 from AvailableTrips  
 where country\_name = countryName and tripDate between dateFrom and dateTo;  
  
 return result;  
end;  
commit;

**Modyfikacja – dodanie pola no\_available\_places**

create or replace function AvailableTripsToChanged(countryName varchar, dateFrom date, dateTo date)  
 return AvailableTripsTable  
as  
 result AvailableTripsTable;  
begin  
 if dateFrom > dateTo then raise\_application\_error(-2000, 'Wrong date given');  
 end if;  
  
 if dateFrom < sysdate then raise\_application\_error(-2000, 'Wrong date given');  
 end if;  
  
 select AvailableTripsO(trip\_id, trip\_name, country\_name, tripDate, availablePlaces)  
 bulk collect into result  
 from AvailableTripsViewChanged  
 where country\_name = countryName and tripDate between dateFrom and dateTo;  
  
 return result;  
end;  
commit;

# Triggery

## Trigger AddReservationTrigger

*Trigger aktualizuje dane w tabeli Log po dodaniu nowej rezerwacji*

create or replace trigger AddReservationTrigger  
 after insert on Reservation  
 for each row  
 begin  
 insert into Log(reservation\_id, log\_date, status)  
 values (:new.reservation\_id, sysdate, :new.status);  
 end;  
commit;

**Modyfikacja po dodaniu pola no\_available\_places**

create or replace trigger AddReservationTrigger  
 after insert on Reservation  
 for each row  
 begin  
 insert into Log(reservation\_id, log\_date, status)  
 values (:new.reservation\_id, sysdate, :new.status);  
  
 update Trip set no\_available\_places = no\_available\_places - 1  
 where trip\_id = :new.trip\_id;  
  
 end;  
commit;

**Trigger ModifyReservationStatusTrigger**

*Trigger aktualizuje dane w tabeli Log po modyfikacji statusu rezerwacji*

create or replace trigger ModifyReservationStatusTrigger  
 after update of status on Reservation  
 for each row  
 begin  
 if :new.status != :old.status then  
 insert into Log(reservation\_id, log\_date, status)  
 values (:new.reservation\_id, sysdate, :new.status);  
 end if;  
 end;  
commit;

**Modyfikacja po dodaniu pola no\_available\_places**

create or replace trigger ModifyReservationStatusTrigger  
 after update of status on Reservation  
 for each row  
 begin  
 if :new.status != :old.status then  
 insert into Log(reservation\_id, log\_date, status)  
 values (:new.reservation\_id, sysdate, :new.status);  
 end if;  
  
 if :new.status = 'C' then  
 update Trip set no\_available\_places = no\_available\_places + 1  
 where trip\_id = :new.trip\_id;  
 end if;  
  
 end;  
commit;

## Trigger BeforeAddingReservationTrigger

*Trigger sprawdza, czy rezerwacja jest możliwa*

create or replace trigger BeforeAddingReservationTrigger  
 before insert on Reservation  
 for each row  
 declare availablePlaces int;  
 begin  
 if not personExistence(:NEW.person\_id) then  
 raise\_application\_error(-20001, 'Person does not exist');  
 end if;  
 if not tripExistence(:new.trip\_id) then  
 raise\_application\_error(-20001, 'Trip does not exist');  
 end if;  
 if not isTheDateCorrect(:new.trip\_id) then  
 raise\_application\_error(-20001, 'The trip has already taken place');  
 end if;  
  
 availablePlaces := getLeftPlaces(:new.trip\_id);  
  
 if availablePlaces = 0 then  
 raise\_application\_error(-20001, 'There are no spots for this trip');  
 end if;  
 end;  
commit;

**Modyfikacja po dodaniu pola no\_available\_places**

create or replace trigger BeforeAddingReservationTrigger  
 before insert on Reservation  
 for each row  
 declare availablePlaces int;  
 begin  
 if not personExistence(:NEW.person\_id) then  
 raise\_application\_error(-20001, 'Person does not exist');  
 end if;  
 if not tripExistence(:new.trip\_id) then  
 raise\_application\_error(-20001, 'Trip does not exist');  
 end if;  
 if not isTheDateCorrect(:new.trip\_id) then  
 raise\_application\_error(-20001, 'The trip has already taken place');  
 end if;  
  
 availablePlaces := getAvailablePlaces(:new.trip\_id);  
  
 if availablePlaces = 0 then  
 raise\_application\_error(-20001, 'There are no spots for this trip');  
 end if;  
 end;  
commit;

## Trigger BeforeChangingReservationStatus

*Trigger sprawdza, czy zmiana statusu rezerwacji jest możliwa*

create or replace trigger BeforeChangingReservationStatus  
 before update of status on Reservation  
 for each row  
 declare  
 pragma autonomous\_transaction;  
 availablePlaces int;  
 begin  
 availablePlaces := getLeftPlaces(:new.trip\_id);  
  
 case :new.status  
 when :old.status then  
 raise\_application\_error(-20001,'Reservation already has this status');  
  
 when 'N' then  
 raise\_application\_error(-20001, 'Cannot change status to NEW');  
  
 when 'C' then  
 null;  
  
 when 'P' then  
 if :old.status = 'C' then  
 if availablePlaces = 0 then  
 raise\_application\_error(-20001, 'Not enough places available');  
 end if;  
 end if;  
  
 else  
 raise\_application\_error(-20001, 'Wrong status given');  
 end case;  
 end;  
commit;

**Modyfikacja po dodaniu pola no\_available\_places**

create or replace trigger BeforeChangingReservationStatusTrigger  
 before update of status on Reservation  
 for each row  
 declare  
 pragma autonomous\_transaction;  
 availablePlaces int;  
 begin  
 availablePlaces := getAvailablePlaces(:new.trip\_id);  
  
 case :new.status  
 when :old.status then  
 raise\_application\_error(-20001,'Reservation already has this status');  
  
 when 'N' then  
 raise\_application\_error(-20001, 'Cannot change status to NEW');  
  
 when 'C' then  
 null;  
  
 when 'P' then  
 if :old.status = 'C' then  
 if availablePlaces = 0 then  
 raise\_application\_error(-20001, 'Not enough places available');  
 end if;  
 end if;  
  
 else  
 raise\_application\_error(-20001, 'Wrong status given');  
 end case;  
 end;  
commit;

## Trigger BeforeChangingMaxPlacesTrigger

*Trigger sprawdza, czy zmiana maksymalnej liczby miejsc jest możliwa*

create or replace trigger BeforeChangingMaxPlacesTrigger  
 before update of max\_no\_places on Trip  
 for each row  
 declare  
 pragma autonomous\_transaction;  
 availablePlaces int;  
 begin  
 availablePlaces := getAvailablePlaces(:new.trip\_id);  
  
 if not tripExistence(:new.trip\_id) then  
 raise\_application\_error(-20001, 'Trip with this id does not exist');  
 end if;  
  
 if :new.max\_no\_places <= 0 then  
 raise\_application\_error(-20001, 'Wrong number of mx places given');  
 end if;  
  
 if :old.max\_no\_places - availablePlaces > :new.max\_no\_places then  
 raise\_application\_error(-20001, 'The amount of booked places is greater than new max\_no\_places value.');  
 end if;  
  
  
 end;  
commit;

# Obiekty

**TripParticipant & TripParticipantsTable**

create or replace type TripParticipant as object(  
 firstname varchar(50),  
 lastname varchar(50),  
 person\_id int,  
 trip\_name varchar(100),  
 trip\_date date  
);  
  
create or replace type TripParticipantsTable is table of TripParticipant;

**PersonReservations0 & PersonReservationsTable**

create or replace type PersonReservationsO as object(  
 trip\_name varchar(100),  
 trip\_date date,  
 trip\_id int,  
 country\_name varchar(50),  
 reservation\_id int,  
 status char(1)  
);  
  
create or replace type PersonReservationsTable is table of PersonReservationsO;

**AvailableTrips0 & AvailableTripsTable**

create or replace type AvailableTripsO as object(  
 trip\_id int,  
 trip\_name varchar(100),  
 country\_name varchar(50),  
 tripDate date,  
 availablePlaces int  
);  
  
create or replace type AvailableTripsTable is table of AvailableTripsO;