

# Sprawozdanie PL/SQL

Kacper Ćwiertnia

## Programowanie proceduralne - widoki, procedury, triggerzy itp.

### I. Oracle PL/SQL

#### 1. Tabele

Trip (trip\_id, name, country, date, no\_places)

Person (person\_id, firstname, lastname)

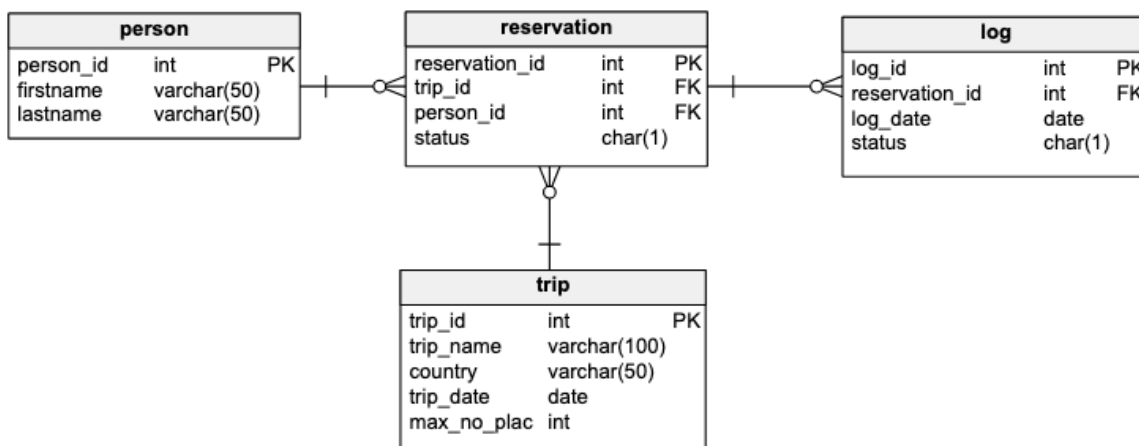
Reservation (reservation\_id, trip\_id, person\_id, status)

Pole status w tabeli Rezerwacje może przyjmować jedną z 4 wartości

N – New

P – Paid



C – Canceled



#### 1. Wypełnianie tabeli przykładowymi danymi

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

## Tabela COUNTRY

	 COUNTRY_ID ↕	 COUNTRY_NAME ↕
1	1	Poland
2	2	Germany
3	3	USA
4	4	China

*Tabela słownikowa zawierająca listę krajów.*

2. Wypełnianie tabeli 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.

## Tabela PERSON

	 PERSON_ID ↕	 FIRSTNAME ↕	 LASTNAME ↕
1	1	Kanye	West
2	2	Donald	Trump
3	3	Mariusz	Pudzianowski
4	4	Adam	Małysz
5	5	Kim	Kardashian
6	6	Zbigniew	Stonoga
7	7	Robert	Lewandowski
8	8	Iga	Świątek
9	9	Cristiano	Ronaldo
10	10	Leo	Messi

*Zawiera listę osób.*

## Tabela RESERVATION

	RESERVATION_ID	TRIP_ID	PERSON_ID	STATUS
1	1	2	1	P
2	2	2	2	C
3	3	2	3	C
4	4	4	4	P
5	5	4	5	C
6	6	4	6	C
7	7	5	7	P
8	8	5	8	N
9	9	6	9	P
10	10	6	10	C
11	21	6	1	P

Zawiera listę rezerwacji wraz z jej danymi: id wycieczki, id osoby rezerwującej i status realizacji.

## Tabela TRIP

	TRIP_ID	TRIP_NAME	COUNTRY_ID	TRIP_DATE	MAX_NO_PLACES
1	2	Polskie góry	1	2022-09-12	3
2	4	Niemieckie browarnictwo	2	2023-03-28	3
3	5	Śladami raperów z Compton	3	2022-11-11	2
4	6	Świątynia porcelany	4	2023-04-11	2
5	21	50 twarzy kremówki	1	2024-03-12	3

Zawiera listę wycieczek wraz z jej danymi: nazwa, id kraju, datę i maksymalną ilość miejsc.

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)
  - b) Trips (country,trip\_date, trip\_name,no\_places, no\_available\_places)
  - c) AvailableTrips (country,trip\_date, trip\_name,no\_places, no\_available\_places)

Proponowany zestaw widoków można rozbudować wedle uznania/potrzeb

- np. można dodać nowe/pomocnicze widoki
- np. można zmienić def. widoków, dodając nowe/potrzebne pola

## Widok RESERVATIONS

```
create or replace view RESERVATIONS
as
    select C.COUNTRY_NAME, T.TRIP_DATE, T.TRIP_NAME, P.FIRSTNAME, P.LASTNAME, R.RESERVATION_ID, R.STATUS
    from RESERVATION R join TRIP T on R.TRIP_ID = T.TRIP_ID
        join PERSON P on R.PERSON_ID = P.PERSON_ID
        join COUNTRY C on T.COUNTRY_ID = C.COUNTRY_ID;
```

Wyświetla informacje o wszystkich rezerwacjach w systemie wraz z jej danymi: kraj, data wycieczki, nazwa wycieczki, imię osoby rezerwującej, nazwisko osoby rezerwującej, id rezerwacji i status realizacji.

### Wynik wywołania

	📄 COUNTRY_NAME	📅 TRIP_DATE	📄 TRIP_NAME	👤 FIRSTNAME	👤 LASTNAME	📄 RESERVATION_ID	📄 STATUS
1	Poland	2022-09-12	Polskie góry	Kanye	West	1	P
2	China	2023-04-11	Świątynia porcelany	Kanye	West	21	P
3	Poland	2022-09-12	Polskie góry	Donald	Trump	2	N
4	Poland	2022-09-12	Polskie góry	Mariusz	Pudzianowski	3	C
5	Germany	2023-03-28	Niemieckie browarnictwo	Adam	Małysz	4	P
6	Germany	2023-03-28	Niemieckie browarnictwo	Kim	Kardashian	5	C
7	Germany	2023-03-28	Niemieckie browarnictwo	Zbigniew	Stonoga	6	C
8	USA	2022-11-11	Śladami raperów z Compton	Robert	Lewandowski	7	P
9	USA	2022-11-11	Śladami raperów z Compton	Iga	Świątek	8	N
10	China	2023-04-11	Świątynia porcelany	Cristiano	Ronaldo	9	P
11	China	2023-04-11	Świątynia porcelany	Leo	Messi	10	C

## Widok pomocniczy NUMOFRESERVATIONS

```
create or replace view NUMOFRESERVATIONS
as
    select T.TRIP_ID, NVL(AC.NUM_OF_RESERVATIONS,0) as NUM_OF_RESERVATIONS
    from (select T.TRIP_ID, count(R.STATUS) as NUM_OF_RESERVATIONS
    from TRIP T join RESERVATION R on T.TRIP_ID = R.TRIP_ID
    where R.STATUS != 'C'
    group by T.TRIP_ID) AC right join TRIP T on T.TRIP_ID = AC.TRIP_ID;
```

Wyświetla informacje o ilości rezerwacji, dla każdej wycieczki.

### Wynik wywołania

	📄 TRIP_ID	📄 NUM_OF_RESERVATIONS
1	6	2
2	2	2
3	4	1
4	5	2
5	21	0

## Widok TRIPS

```
create or replace view TRIPS
as
select T.TRIP_ID, C.COUNTRY_NAME, T.TRIP_DATE, T.TRIP_NAME, T.MAX_NO_PLACES, (T.MAX_NO_PLACES - AC.NUM_OF_RESERVATIONS) as NO_AVALIABLE_PLACES
from NUMOFRESERVATIONS AC join TRIP T on AC.TRIP_ID = T.TRIP_ID
join COUNTRY C on T.COUNTRY_ID = C.COUNTRY_ID;
```

Wyświetla informacje o wszystkich wycieczkach i ilości wolnych miejsc.

### Wynik wywołania

	TRIP_ID	COUNTRY_NAME	TRIP_DATE	TRIP_NAME	MAX_NO_PLACES	NO_AVALIABLE_PLACES
1	6	China	2023-04-11	Świątynia porcelany	2	0
2	2	Poland	2022-09-12	Polskie góry	3	1
3	4	Germany	2023-03-28	Niemieckie browarnictwo	3	2
4	5	USA	2022-11-11	Śladami raperów z Compton	2	0
5	21	Poland	2024-03-12	50 twarzy kremówki	10	10

## Widok AVAILIABLETRIPS

```
create or replace view AVAILIABLETRIPS
as
select *
from TRIPS T
where T.TRIP_DATE > CURRENT_DATE and T.NO_AVALIABLE_PLACES > 0;
```

Wyświetla informacje o wszystkich nieodbytych jeszcze wycieczkach i ilości wolnych miejsc.

### Wynik wywołania

	COUNTRY_NAME	TRIP_DATE	TRIP_NAME	MAX_NO_PLACES	NO_AVALIABLE_PLACES
1	Germany	2023-03-28	Niemieckie browarnictwo	3	2
2	Poland	2024-03-12	50 twarzy kremówki	10	10

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
  - b) PersonReservations (person\_id), procedura ma zwracać podobny zestaw danych jak widok Reservations
  - c) AvailableTrips (country, date\_from, date\_to)

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.

Typ TRIP\_PARTICIPANT

```
create type trip_participant as OBJECT
(
    reservation_id int,
    trip_id int,
    person_id int,
    firstname varchar2(50),
    lastname varchar2(50)
)
```

*Typ, który będzie zwracany w wyniku wywołania F\_TRIP\_PARTICIPANTS i F\_PERSON\_RESERVATIONS.*

Typ TRIP\_PARTICIPANT\_TABLE

```
create type TRIP_PARTICIPANT_TABLE as table of TRIP_PARTICIPANT
```

*Tabela, która będzie zwracana w wyniku wywołania F\_TRIP\_PARTICIPANTS i F\_PERSON\_RESERVATIONS.*

Procedura TRIP\_EXIST

```
create procedure TRIP_EXIST(t_id in TRIP.TRIP_ID%type)
as
    tmp char(1);
begin
    select 1 into tmp from TRIP where TRIP_ID = t_id;
exception
    when NO_DATA_FOUND then
        raise_application_error(-20001, 'TRIP NOT FOUND');
end;
```

*Sprawdza czy wycieczka o danym id istnieje.*

## Funkcja F\_TRIP\_PARTICIPANTS

```
create function F_TRIP_PARTICIPANTS(trip_id int)
    return TRIP_PARTICIPANT_TABLE
as
    result TRIP_PARTICIPANT_TABLE;
begin
    TRIP_EXIST( T_ID: trip_id);
    select TRIP_PARTICIPANT(R.RESERVATION_ID, R.TRIP_ID, P.PERSON_ID, P.FIRSTNAME, P.LASTNAME) bulk collect into result
    from RESERVATION R join PERSON P on R.PERSON_ID = P.PERSON_ID
    where R.TRIP_ID = F_TRIP_PARTICIPANTS.trip_id and R.STATUS != 'C';
    return result;
end;
```

Zwraca informacje o rezerwacjach danej wycieczki, które nie mają statusu 'C'.

### Wynik wywołania

	RESERVATION_ID	TRIP_ID	PERSON_ID	FIRSTNAME	LASTNAME
1	1	2	1	Kanye	West

Tylko jedna rezerwacja została wyświetlona, ponieważ dwie pozostałe mają status 'C'.

## Procedura PERSON\_EXIST

```
create or replace procedure PERSON_EXIST(p_id in PERSON.PERSON_ID%type)
as
    tmp char(1);
begin
    select 1 into tmp from PERSON where PERSON_ID = p_id;
exception
    when NO_DATA_FOUND then
        raise_application_error(-20001, 'PERSON NOT FOUND');
end;
```

Sprawdza czy osoba o danym id istnieje.

## Funkcja F\_PERSON\_RESERVATIONS

```
create or replace function F_PERSON_RESERVATIONS(person_id int)
    return TRIP_PARTICIPANT_TABLE
as
    result TRIP_PARTICIPANT_TABLE;
begin
    PERSON_EXIST( p_id: person_id);
    select TRIP_PARTICIPANT(R.RESERVATION_ID, R.TRIP_ID, P.PERSON_ID, P.FIRSTNAME, P.LASTNAME) bulk collect into result
    from RESERVATION R join PERSON P on R.PERSON_ID = P.PERSON_ID
    where R.PERSON_ID = F_PERSON_RESERVATIONS.person_id;
    return result;
end;
```

Zwraca dane rezerwacji osoby o podanym id.

### Wynik wywołania

	RESERVATION_ID	TRIP_ID	PERSON_ID	FIRSTNAME	LASTNAME
1	2	2	2	Donald	Trump

## Typ TRIP\_INFO

```
create type trip_info as OBJECT
(
    country varchar2(50),
    trip_date date,
    trip_name varchar2(100),
    max_no_places int,
    no_available_places int
)
```

Typ, który będzie zwracany w wyniku wywołania *F\_AVAILABLE\_TRIPS*.

## Typ TRIP\_INFO\_TABLE

```
create type TRIP_INFO_TABLE as table of TRIP_INFO;
```

Typ, który będzie zwracany w wyniku wywołania *F\_AVAILABLE\_TRIPS*.

## Procedura COUNTRY\_EXIST

```
create procedure COUNTRY_EXIST(c_name in COUNTRY.COUNTRY_NAME%type)
as
    tmp char(1);
begin
    select 1 into tmp from COUNTRY where COUNTRY_NAME = c_name;
exception
    when NO_DATA_FOUND then
        raise_application_error(-20001, 'COUNTRY NOT FOUND');
end;
```

Sprawdza czy kraj o danej nazwie istnieje.

## Funkcja F\_AVAILABLE\_TRIPS

```
create or replace function F_AVAILABLE_TRIPS(country varchar, date_from date, date_to date)
return TRIP_INFO_TABLE
as
    result TRIP_INFO_TABLE;
begin
    COUNTRY_EXIST(C_NAME country);

    select TRIP_INFO(AV.COUNTRY_NAME, AV.TRIP_DATE, AV.TRIP_NAME, AV.MAX_NO_PLACES, AV.NO_AVAILABLE_PLACES) bulk collect into result
    from AVAILABLETRIPS AV
    where AV.COUNTRY_NAME = F_AVAILABLE_TRIPS.country and AV.TRIP_DATE >= F_AVAILABLE_TRIPS.date_from and AV.TRIP_DATE <= F_AVAILABLE_TRIPS.date_to;
    return result;
end;
```

Zwraca dane wycieczki do podanego kraju w podanym zakresie czasu.

## Wynik przykładowego wywołania

	COUNTRY	TRIP_DATE	TRIP_NAME	MAX_NO_PLACES	NO_AVAILABLE_PLACES
1	Germany	2023-03-28	Niemieckie browarnictwo	3	2



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 są wolne miejsca
  - 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)
  - c) ModifyNoPlaces (trip\_id, no\_places), nie wszystkie zmiany liczby miejsc są dozwolone, nie można zmniejszyć liczby miejsc na wartość poniżej liczby zarezerwowanych miejsc

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...)

## Procedura ADD\_RESERVATION

```
create or replace procedure ADD_RESERVATION(p_trip_id int, p_person_id int)
as
    tmp char(1);
begin
    TRIP_EXIST( T_ID: p_trip_id);
    PERSON_EXIST( P_ID: p_person_id);

    select 1 into tmp from AVAILABLETRIPS
    where TRIP_ID = p_trip_id;

    insert into RESERVATION(TRIP_ID, PERSON_ID, STATUS) VALUES (p_trip_id, p_person_id, 'N');

exception
    when NO_DATA_FOUND then
        raise_application_error(-20001, 'TRIP HAD TAKEN PLACE OR IS FULLY BOOKED');

end;
```

*Dodaje rezerwację na podstawie id wycieczki i osoby, procedura najpierw sprawdza czy istnieje wycieczka i osoba o podanym id, a następnie czy wycieczka ma wolne miejsce i czy się jeszcze nie odbyła.*

## Procedura RESERVATION\_EXIST

```
create procedure RESERVATION_EXIST(r_id RESERVATION.RESERVATION_ID%type)
as
    tmp char(1);
begin
    select 1 into tmp from RESERVATION where RESERVATION_ID = r_id;
exception
    when NO_DATA_FOUND then
        raise_application_error(-20001, 'RESERVATION NOT FOUND');
end;
```

*Sprawdza czy rezerwacja o podanym id istnieje.*

## Procedura MODIFY\_RESERVATION\_STATUS

```
create or replace procedure MODIFY_RESERVATION_STATUS(p_reservation_id int, p_status char)
as
    r_status char(1);
    r_places int;
begin
    RESERVATION_EXIST(p_reservation_id);

    if p_status not in ('N', 'P', 'C') then
        raise_application_error(-20002, 'WRONG STATUS');
    end if;

    select STATUS into r_status from RESERVATION where RESERVATION_ID = p_reservation_id;
    select T.NO_AVAILABLE_PLACES into r_places from TRIPS T join RESERVATION R on T.TRIP_ID = R.TRIP_ID where RESERVATION_ID = p_reservation_id;

    if(p_status = 'C' or p_status = 'P' and r_status = 'N' or p_status = 'P' and r_status = 'C' and r_places > 0 or p_status = 'N' and r_status = 'C' and r_places > 0) then
        update RESERVATION set STATUS = p_status where RESERVATION_ID = p_reservation_id;
    else
        raise_application_error(-20002, 'CANNOT CHANGE STATUS');
    end if;
end;
```

*Zmienia status rezerwacji o podanym id na inny, procedura sprawdza czy rezerwacja istnieje i czy można dokonać zmiany statusu.*

## Procedura MODIFY\_NO\_PLACES

```
create procedure MODIFY_NO_PLACES(p_trip_id int, p_no_places int)
as
    r_places int;
begin
    TRIP_EXIST(p_trip_id);

    select NUM_OF_RESERVATIONS into r_places from NUMOFRESERVATIONS where TRIP_ID = p_trip_id;

    if(r_places <= p_no_places)then
        update TRIP set MAX_NO_PLACES = p_no_places where TRIP_ID = p_trip_id;
    else
        raise_application_error(-20002, 'TOO MANY RESERVATIONS');
    end if;
end;
```

*Zmienia maksymalną ilość miejsc danej wycieczki, procedura sprawdza czy dana wycieczka istnieje i czy nie ma już za dużo rezerwacji.*

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

## Procedura ADD\_RESERVATION

```
create or replace procedure ADD_RESERVATION(p_trip_id int, p_person_id int)
as
    tmp char(1);
    p_reservation_id int;
begin
    TRIP_EXIST( T_ID: p_trip_id);
    PERSON_EXIST( P_ID: p_person_id);

    select 1 into tmp from AVAILBLETRIPS
    where TRIP_ID = p_trip_id;

    insert into RESERVATION(TRIP_ID, PERSON_ID, STATUS) VALUES (p_trip_id, p_person_id, 'N') returning RESERVATION_ID into p_reservation_id;
    insert into LOG(RESERVATION_ID, LOG_DATE, STATUS) VALUES (p_reservation_id, current_date, 'N');

exception
    when NO_DATA_FOUND then
        raise_application_error(-20001, 'TRIP HAD TAKEN PLACE OR IS FULLY BOOKED');

end;
```

*Zmodyfikowana procedura, która przy dodawaniu rezerwacji dodaje informacje o nowej rezerwacji do logów.*

## Procedura MODIFY\_RESERVATION\_STATUS

```
create procedure MODIFY_RESERVATION_STATUS(p_reservation_id int, p_status char)
as
    r_status char(1);
    r_places int;
begin
    RESERVATION_EXIST( R_ID: p_reservation_id);

    if p_status not in ('N','P','C') then
        raise_application_error(-20002, 'WRONG STATUS');
    end if;

    select STATUS into r_status from RESERVATION where RESERVATION_ID = p_reservation_id;
    select T.NO_AVAILBLE_PLACES into r_places from TRIPS T join RESERVATION R on T.TRIP_ID = R.TRIP_ID where RESERVATION_ID = p_reservation_id;

    if(p_status = 'C' or p_status = 'P' and r_status = 'N' or p_status = 'P' and r_status = 'C' and r_places > 0 or p_status = 'N' and r_status = 'C' and r_places > 0) then
        update RESERVATION set STATUS = p_status where RESERVATION_ID = p_reservation_id;
        insert into LOG(RESERVATION_ID, LOG_DATE, STATUS) values (p_reservation_id, current_date, p_status);
    else
        raise_application_error(-20002, 'CANNOT CHANGE STATUS');
    end if;

end;
```

*Zmodyfikowana procedura, która przy zmianie statusu dodaje informacje o zmianie do logów.*

### 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 triggerów

- trigger obsługujący dodanie rezerwacji
- trigger obsługujący zmianę statusu
- trigger zabraniający usunięcia rezerwacji

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

## Trigger TR\_UPDATE\_LOG

```
create or replace trigger TR_UPDATE_LOG
  after insert or update
  on RESERVATION
  for each row
begin
  insert into LOG(RESERVATION_ID, LOG_DATE, STATUS) VALUES(:new.RESERVATION_ID, current_date, :new.STATUS);
end;
```

*Dodaje do logów informacje o dodaniu rezerwacji lub zmiany jej statusu.*

## Procedura ADD\_RESERVATION\_2

```
create or replace procedure ADD_RESERVATION_2(p_trip_id int, p_person_id int)
as
  tmp char(1);
begin
  TRIP_EXIST( T_ID: p_trip_id);
  PERSON_EXIST( P_ID: p_person_id);

  select 1 into tmp from AVAILABLETRIPS
  where TRIP_ID = p_trip_id;

  insert into RESERVATION(TRIP_ID, PERSON_ID, STATUS) VALUES (p_trip_id, p_person_id, 'N');

exception
  when NO_DATA_FOUND then
    raise_application_error(-20001, 'TRIP HAD TAKEN PLACE OR IS FULLY BOOKED');

end;
```

*Zmieniona procedura ADD\_RESERVATION, przy jej wywołaniu zostanie uruchomiony trigger TR\_UPDATE\_LOG.*

## Procedura MODIFY\_RESERVATION\_STATUS\_2

```
create or replace procedure MODIFY_RESERVATION_STATUS_2(p_reservation_id int, p_status char)
as
  r_status char(1);
  r_places int;
begin
  RESERVATION_EXIST( R_ID: p_reservation_id);

  if p_status not in ('N', 'P', 'C') then
    raise_application_error(-20002, 'WRONG STATUS');
  end if;

  select STATUS into r_status from RESERVATION where RESERVATION_ID = p_reservation_id;
  select T.NO_AVAILABLE_PLACES into r_places from TRIPS T join RESERVATION R on T.TRIP_ID = R.TRIP_ID where RESERVATION_ID = p_reservation_id;

  if(p_status = 'C' or p_status = 'P' and r_status = 'N' or p_status = 'P' and r_status = 'C' and r_places > 0 or p_status = 'N' and r_status = 'C' and r_places > 0) then
    update RESERVATION set STATUS = p_status where RESERVATION_ID = p_reservation_id;
  else
    raise_application_error(-20002, 'CANNOT CHANGE STATUS');
  end if;
end;
```

*Zmieniona procedura MODIFY\_RESERVATION\_STATUS, przy jej wywołaniu zostanie uruchomiony trigger TR\_UPDATE\_LOG.*

### Trigger TR\_DELETE\_RESERVATION

```
create or replace trigger TR_DELETE_RESERVATION
before delete
on RESERVATION
for each row
begin
    raise_application_error(-20001, 'CANNOT REMOVE RESERVATION');
end;
```

*Zapobiega usunięciu rezerwacji.*

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

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

trigger obsługujący dodanie rezerwacji

trigger obsługujący zmianę statusu

Oczywiście po wprowadzeniu tej zmiany należy uaktualnić procedury modyfikujące dane.

Najlepiej to zrobić tworząc nowe wersje (np. z sufiksem 3)

### Trigger TR\_ADD\_RESERVATION

```
create or replace trigger TR_ADD_RESERVATION
before insert
on RESERVATION
for each row
declare
    tmp char;
begin
    TRIP_EXIST( T_ID: :new.TRIP_ID);
    PERSON_EXIST( P_ID: :new.PERSON_ID);

    select 1 into tmp from AVAILABLETRIPS
    where TRIP_ID = :new.TRIP_ID;

    exception
    when NO_DATA_FOUND then
        raise_application_error(-20001, 'TRIP HAD TAKEN PLACE OR IS FULLY BOOKED');
end;
```

*Przed dodaniem rezerwacji sprawdza czy wycieczka istnieje, czy osoba istnieje, a na końcu sprawdza czy wycieczka ma jeszcze wolne miejsca.*

## Procedura ADD\_RESERVATION\_3

```
create or replace procedure ADD_RESERVATION_3(p_trip_id int, p_person_id int)
as
begin
    insert into RESERVATION(TRIP_ID, PERSON_ID, STATUS) VALUES (p_trip_id, p_person_id, 'N');
end;
```

Zmieniona procedura ADD\_RESERVATION\_2, przed dodaniem rezerwacji zostanie wywołany trigger TR\_ADD\_RESERVATION

## Trigger TR\_UPDATE\_RESERVATION

```
create trigger TR_UPDATE_RESERVATION
before update
on RESERVATION
for each row
declare
    r_places int;
begin
    if :new.STATUS not in ('N','P','C') then
        raise_application_error(-20002, 'WRONG STATUS');
    end if;

    select NO_AVAILABLE_PLACES into r_places from TRIP where TRIP_ID = :old.TRIP_ID;

    if not(:new.STATUS = 'C' or :new.STATUS = 'P' and :old.STATUS = 'N' or :new.STATUS = 'P' and :old.STATUS = 'C' and r_places > 0 or :new.STATUS = 'N' and :old.STATUS = 'C' and r_places > 0) then
        raise_application_error(-20002, 'CANNOT CHANGE STATUS');
    end if;

    exception
    when NO_DATA_FOUND then
        raise_application_error(-20001, 'TRIP HAD TAKEN PLACE OR IS FULLY BOOKED');
end;
```

Sprawdza, czy można dokonać zmiany statusu rezerwacji.

## Procedura MODIFY\_RESERVATION\_STATUS\_3

```
create procedure MODIFY_RESERVATION_STATUS_3(p_reservation_id int, p_status char)
as
begin
    RESERVATION_EXIST( R_ID: p_reservation_id);

    update RESERVATION set STATUS = p_status where RESERVATION_ID = p_reservation_id;
end;
```

Zmienia status rezerwacji.

9. Zmiana struktury bazy danych, w tabeli wycieczki dodajemy redundantne pole no\_available\_places

Obsługa redundantnego pola (kontrola liczby dostępnych miejsc) w procedurach

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

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

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)

## Tabela TRIP

	TRIP_ID	TRIP_NAME	COUNTRY_ID	TRIP_DATE	MAX_NO_PLACES	NO_AVAILABLE_PLACES
1	2	Polskie góry	1	2022-09-12	3	1
2	4	Niemieckie browarstwo	2	2023-03-28	3	2
3	5	Śladami raperów z Compton	3	2022-11-11	2	0
4	6	Świątynia porcelany	4	2023-04-11	2	0
5	21	50 twarzy kremówki	1	2024-03-12	10	10

## Procedura RECALC

```
create or replace procedure RECALC(p_trip_id int)
as
    r_places int;
begin
    TRIP_EXIST( T_ID: p_trip_id);

    select NUM_OF_RESERVATIONS into r_places from NUMOFRESERVATIONS where TRIP_ID = p_trip_id;

    update TRIP set NO_AVAILABLE_PLACES = MAX_NO_PLACES - r_places where TRIP_ID = p_trip_id;
end;
```

Aktualizuje ilość dostępnych miejsc podanej wycieczki.

## Widok TRIPS\_4

```
create or replace view TRIPS_2
as
    select T.TRIP_ID, C.COUNTRY_NAME, T.TRIP_DATE, T.TRIP_NAME, T.MAX_NO_PLACES, T.NO_AVAILABLE_PLACES
    from TRIP T join COUNTRY C on T.COUNTRY_ID = C.COUNTRY_ID
```

Zmieniona wersja widoku TRIPS, która korzysta z pola NO\_AVAILABLE\_PLACES tabeli TRIP.

## Widok AVAILABLETRIPS\_4

```
create or replace view AVAILABLETRIPS_2
as
    select T.TRIP_ID, T.COUNTRY_NAME, T.TRIP_DATE, T.TRIP_NAME, T.MAX_NO_PLACES, T.NO_AVAILABLE_PLACES
    from TRIPS T
    where T.TRIP_DATE > CURRENT_DATE and T.NO_AVAILABLE_PLACES > 0
```

Zmieniona wersja widoku AVAILABLETRIPS, która korzysta z pola NO\_AVAILABLE\_PLACES tabeli TRIP.

## Procedura ADD\_RESERVATION\_4

```
create procedure ADD_RESERVATION_4(p_trip_id int, p_person_id int)
as
begin
    insert into RESERVATION(TRIP_ID, PERSON_ID, STATUS) VALUES (p_trip_id, p_person_id, 'N');
    RECALC( P_TRIP_ID: p_trip_id);
end;
```

Dodaję nową rezerwację i aktualizuję pole NO\_AVAILABLE\_PLACES w tabeli TRIP przy pomocy procedury RECALC.

## Funkcja F\_AVAILABLE\_TRIPS\_4

```
create or replace function F_AVAILABLE_TRIPS_4(country varchar, date_from date, date_to date)
return TRIP_INFO_TABLE
as
result TRIP_INFO_TABLE;
begin
COUNTRY_EXIST( C_NAME: country);

select TRIP_INFO(C.COUNTRY_NAME, T.TRIP_DATE, T.TRIP_NAME, T.MAX_NO_PLACES, T.NO_AVAILABLE_PLACES) bulk collect into result
from TRIP T join COUNTRY C on C.COUNTRY_ID = T.COUNTRY_ID
where C.COUNTRY_NAME = F_AVAILABLE_TRIPS_4.country and T.TRIP_DATE >= current_date and T.TRIP_DATE >= F_AVAILABLE_TRIPS_4.date_from and T.TRIP_DATE <= F_AVAILABLE_TRIPS_4.date_to;
return result;
end;
```

Zwraca dane wycieczki do podanego kraju w podanym zakresie czasu, korzysta z pola NO\_AVAILABLE\_PLACES tabeli TRIP.

## Procedura MODIFY\_NO\_PLACES\_4

```
create procedure MODIFY_NO_PLACES_4(p_trip_id int, p_no_places int)
as
r_places int;
m_places int;
begin
TRIP_EXIST( T_ID: p_trip_id);

select NO_AVAILABLE_PLACES into r_places from TRIP where TRIP_ID = p_trip_id;
select MAX_NO_PLACES into m_places from TRIP where TRIP_ID = p_trip_id;

if(m_places-r_places <= p_no_places) then
update TRIP set MAX_NO_PLACES = p_no_places where TRIP_ID = p_trip_id;
RECALC( P_TRIP_ID: p_trip_id);
else
raise_application_error(-20002,'TOO MANY RESERVATIONS');
end if;
end;
```

Sprawdza czy podana ilość miejsc nie jest mniejsza od ilości rezerwacji, jeżeli nie to zmienia ilość miejsc wycieczki, a następnie przy pomocy RECALC aktualizuję ilość wolnych miejsc.



## Procedura MODIFY\_RESERVATION\_STATUS\_4

```
create procedure MODIFY_RESERVATION_STATUS_4(p_reservation_id int, p_status char)
as
    tmp char;
    trip int;
begin
    RESERVATION_EXIST( R_ID: p_reservation_id);

    select STATUS into tmp from RESERVATION where RESERVATION_ID = p_reservation_id;
    select TRIP_ID into trip from RESERVATION where RESERVATION_ID = p_reservation_id;

    update RESERVATION set STATUS = p_status where RESERVATION_ID = p_reservation_id;

    RECALC( P_TRIP_ID: trip);
end;
```

*Zmienia status rezerwacji oraz przy pomocy procedury RECACAL aktualizuję ilość wolnych miejsc.*

10. Zmiana strategii obsługi redundantnego pola no\_available\_places, realizacja przy pomocy triggerów

trigger obsługujący dodanie rezerwacji

trigger obsługujący zmianę statusu

trigger obsługujący zmianę liczby miejsc na poziomie wycieczki

Oczywiście po wprowadzeniu tej zmiany należy uaktualnić procedury modyfikujące dane.

Najlepiej to zrobić tworząc nowe wersje (np. z sufiksem 5)

## Trigger TR\_UPDATE\_PLACES

```
create trigger TR_UPDATE_PLACES
after update
on TRIP
for each row
begin
    RECALC( P_TRIP_ID: TRIP_ID);
end;
```

Aktualizuję ilość wolnych miejsc wycieczki po edycji tabeli TRIP.

Trigger TR\_UPDATE\_PLACES\_2

```
create trigger TR_UPDATE_PLACES_2
after insert or update
on RESERVATION
for each row
begin
    RECALC( P_TRIP_ID: :new.TRIP_ID);
end;
```

Aktualizuję ilość wolnych miejsc wycieczki po edycji tabeli RESERVATION.

Procedura ADD\_RESERVATION\_5

```
create or replace procedure ADD_RESERVATION_5(p_trip_id int, p_person_id int)
as
begin
    insert into RESERVATION(TRIP_ID, PERSON_ID, STATUS) VALUES (p_trip_id, p_person_id, 'N');
end;
```

Dodaje rezerwację po czym trigger TR\_UPDATE\_PLACES\_2 zaktualizuje ilość wolnych miejsc.

Procedura MODIFY\_NO\_PLACES\_5

```
create procedure MODIFY_NO_PLACES_5(p_trip_id int, p_no_places int)
as
    r_places int;
    m_places int;
begin
    TRIP_EXIST( T_ID: p_trip_id);

    select NO_AVAILABLE_PLACES into r_places from TRIP where TRIP_ID = p_trip_id;
    select MAX_NO_PLACES into m_places from TRIP where TRIP_ID = p_trip_id;

    if(m_places-r_places <= p_no_places) then
        update TRIP set MAX_NO_PLACES = p_no_places where TRIP_ID = p_trip_id;
    end if;
end;
```

Zmienia ilość miejsc wycieczki po czym trigger TR\_UPDATE\_PLACES zaktualizuje ilość wolnych miejsc.

## Procedura MODIFY\_RESERVATION\_STATUS\_5

```
create or replace procedure MODIFY_RESERVATION_STATUS_5(p_reservation_id int, p_status char)
as
    tmp char;
    trip int;
begin
    RESERVATION_EXIST( R_ID: p_reservation_id);

    select STATUS into tmp from RESERVATION where RESERVATION_ID = p_reservation_id;
    select TRIP_ID into trip from RESERVATION where RESERVATION_ID = p_reservation_id;

    update RESERVATION set STATUS = p_status where RESERVATION_ID = p_reservation_id;
end;
```

*Zmienia status rezerwacji po czym trigger TR\_UPDATE\_PLACES\_2 zaktualizuje ilość wolnych miejsc.*

## 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