Oracle PL/Sql

widoki, funkcje, procedury, triggery ćwiczenie

Imiona i nazwiska autorów : Filip Węgrzyn, Seweryn Tasior

Tabele

- `Trip` - wycieczki
  - `trip_id` - identyfikator, klucz główny
  - `trip_name` - nazwa wycieczki
  - `country` - nazwa kraju
  - `trip_date` - data
  - `max_no_places` - maksymalna liczba miejsc na wycieczkę
- `Person` - osoby
  - `person_id` - identyfikator, klucz główny
  - `firstname` - imię
  - `lastname` - nazwisko
- `Reservation` - rezerwacje/bilety na wycieczkę
  - `reservation_id` - identyfikator, klucz główny
  - `trip_id` - identyfikator wycieczki
  - `person_id` - identyfikator osoby
  - `status` - status rezerwacji
    - `N` – New - Nowa
    - `P` – Confirmed and Paid – Potwierdzona i zapłacona
    - `C` – Canceled - Anulowana
- `Log` - dziennik zmian statusów rezerwacji
  - `log_id` - identyfikator, klucz główny
  - `reservation_id` - identyfikator rezerwacji
  - `log_date` - data zmiany
  - `status` - status

```
create sequence s_person_seq
   start with 1
```

```
   increment by 1;

create table person
(
  person_id int not null
      constraint pk_person
         primary key,
  firstname varchar(50),
  lastname varchar(50)
)

alter table person
    modify person_id int default s_person_seq.nextval;
```

```
create sequence s_trip_seq
   start with 1
   increment by 1;

create table trip
(
  trip_id int  not null
     constraint pk_trip
         primary key,
  trip_name varchar(100),
  country varchar(50),
  trip_date date,
  max_no_places int
);

alter table trip
    modify trip_id int default s_trip_seq.nextval;
```

```
create sequence s_reservation_seq
   start with 1
   increment by 1;

create table reservation
(
  reservation_id int not null
      constraint pk_reservation
         primary key,
  trip_id int,
  person_id int,
  status char(1)
);

alter table reservation
    modify reservation_id int default s_reservation_seq.nextval;
```

```
alter table reservation
add constraint reservation_fk1 foreign key
( person_id ) references person ( person_id );

alter table reservation
add constraint reservation_fk2 foreign key
( trip_id ) references trip ( trip_id );

alter table reservation
add constraint reservation_chk1 check
(status in ('N','P','C'));
```

```
create sequence s_log_seq
   start with 1
   increment by 1;


create table log
(
    log_id int not null
         constraint pk_log
         primary key,
    reservation_id int not null,
    log_date date not null,
    status char(1)
);

alter table log
    modify log_id int default s_log_seq.nextval;

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 );
```

Dane

Należy wypełnić 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.

```
-- trip
insert into trip(trip_name, country, trip_date, max_no_places)
values ('Wycieczka do Paryza', 'Francja', to_date('2023-09-12', 'YYYY-MM-DD'), 3);

insert into trip(trip_name, country, trip_date,  max_no_places)
values ('Piekny Krakow', 'Polska', to_date('2025-05-03','YYYY-MM-DD'), 2);

insert into trip(trip_name, country, trip_date,  max_no_places)
values ('Znow do Francji', 'Francja', to_date('2025-05-01','YYYY-MM-DD'), 2);

insert into trip(trip_name, country, trip_date,  max_no_places)
values ('Hel', 'Polska', to_date('2025-05-01','YYYY-MM-DD'),  2);

-- person
insert into person(firstname, lastname)
values ('Jan', 'Nowak');

insert into person(firstname, lastname)
values ('Jan', 'Kowalski');

insert into person(firstname, lastname)
values ('Jan', 'Nowakowski');

insert into person(firstname, lastname)
values  ('Novak', 'Nowak');

-- reservation
-- trip1
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 (2, 4, 'P');
```

proszę pamiętać o zatwierdzeniu transakcji

Zadanie 0 - modyfikacja danych, transakcje

Należy zmodyfikować model danych tak żeby rezerwacja mogła dotyczyć kilku miejsc/biletów na wycieczkę

- do tabeli reservation należy dodać pole
  - no_tickets
- do tabeli log należy dodac pole
  - no_tickets

Należy zmodyfikować zestaw danych testowych

Należy przeprowadzić kilka eksperymentów związanych ze wstawianiem, modyfikacją i usuwaniem danych oraz wykorzystaniem transakcji

Skomentuj dzialanie transakcji. Jak działa polecenie `commit`, `rollback`?. Co się dzieje w przypadku wystąpienia błędów podczas wykonywania transakcji? Porównaj sposób programowania operacji wykorzystujących transakcje w Oracle PL/SQL ze znanym ci systemem/językiem MS Sqlserver T-SQL

pomocne mogą być materiały dostępne tu: https://upel.agh.edu.pl/mod/folder/view.php?id=311899 w szczególności dokument: `1_ora_modyf.pdf`

```
ALTER TABLE RESERVATION
    ADD no_tickets int default 0 NOT NULL;
ALTER TABLE LOG
    ADD no_tickets int default 0 NOT NULL;
-- inserting data
-- trip
insert into trip(trip_name, country, trip_date, max_no_places)
values ('Wycieczka do Paryza', 'Francja', to_date('2023-09-12', 'YYYY-MM-DD'), 3);
insert into trip(trip_name, country, trip_date, max_no_places)
values ('Piekny Krakow', 'Polska', to_date('2025-05-03','YYYY-MM-DD'), 2);
insert into trip(trip_name, country, trip_date, max_no_places)
values ('Znow do Francji', 'Francja', to_date('2025-05-01','YYYY-MM-DD'), 2);
insert into trip(trip_name, country, trip_date, max_no_places)
values ('Hel', 'Polska', to_date('2025-05-01','YYYY-MM-DD'), 2);
-- person
insert into person(firstname, lastname)
```

```
values ('Jan', 'Nowak');
insert into person(firstname, lastname)
values ('Jan', 'Kowalski');
insert into person(firstname, lastname)
values ('Jan', 'Nowakowski');
insert into person(firstname, lastname)
values ('Novak', 'Nowak');
-- reservation
-- trip1
insert into reservation(trip_id, person_id, status, no_tickets)
values (1, 1, 'P',1);
2025-03-05
insert into reservation(trip_id, person_id, status, no_tickets)
values (1, 2, 'N',2);
-- trip 2
insert into reservation(trip_id, person_id, status, no_tickets)
values (2, 1, 'P',1);
insert into reservation(trip_id, person_id, status, no_tickets)
values (2, 4, 'C',4);
-- trip 3
insert into reservation(trip_id, person_id, status, no_tickets)
values (2, 4, 'P',3)
```

Działanie transakcji

Transakcje korzystająca ze zasady ACID. Każdy wykonany operacja jest domyślnie transakcją i musi być zatwierdzony przez komendę commit. Jeśli przy jakimś wyowłaniu wyrażenia nastąpi błąd, cała transkacja domyślnie zostanie cofnięta przez rollback, cofając wszystkie zmiany od początku transkacji. W TSQL transakcje są jawnie deklarowane za pomocą BEGIN TRANSACTION, a w przypadku błędu należy wykonać rollback w bloku try catch. W MS SQL Server również transakcje nie są automatyczne i wymagają jawnego rozpoczęcia.

Zadanie 1 - widoki

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

Widoki:

- `vw_reservation`
  - widok łączy dane z tabel: `trip`, `person`, `reservation`
  - zwracane dane: `reservation_id`, `country`, `trip_date`, `trip_name`, `firstname`, `lastname`, `status`, `trip_id`, `person_id`, `no_tickets`
- `vw_trip`
  - widok pokazuje liczbę wolnych miejsc na każdą wycieczkę

    - zwracane dane: `trip_id`, `country`, `trip_date`, `trip_name`, `max_no_places`, `no_available_places` (liczba wolnych miejsc)
- `vw_available_trip`
    - podobnie jak w poprzednim punkcie, z tym że widok pokazuje jedynie dostępne wycieczki (takie które są w przyszłości i są na nie wolne miejsca)

Proponowany zestaw widoków można rozbudować wedle uznania/potrzeb

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

Zadanie 1 - rozwiązanie

```
--widok 1
create or replace view vw_reservation
as
    select R.RESERVATION_ID, COUNTRY, TRIP_DATE, TRIP_NAME, FIRSTNAME, LASTNAME,
R.STATUS, R.TRIP_ID, R.PERSON_ID, R.NO_TICKETS
    from RESERVATION R
    join PERSON P on R.PERSON_ID = P.PERSON_ID
    join TRIP T on R.TRIP_ID = T.TRIP_ID;


--widok 2
create or replace view vw_trip
as
    select TRIP_ID, COUNTRY, TRIP_DATE, TRIP_NAME, MAX_NO_PLACES,
f_get_available_places(TRIP_ID) as NO_AVAILABLE_PLACES
    from TRIP;


--widok 3
create or replace view vw_available_trip
as
    select * from vw_trip
    where NO_AVAILABLE_PLACES>0 and TRIP_DATE>CURRENT_DATE ;


--funkcja pomocnicza obliczająca liczbę wolnych miejsc
create or replace function f_get_available_places(trip_id in TRIP.TRIP_ID%type)
    return number
as
    no_available_places number;
begin
    select MAX_NO_PLACES - (select COALESCE(sum(NO_TICKETS), 0)
                            from RESERVATION R
                            where R.TRIP_ID = trip_id and R.STATUS<>'C')
    into no_available_places
    from TRIP T
```

```
    where T.TRIP_ID = trip_id;

    return no_available_places;
end;
```

Zadanie 2 - funkcje

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

Procedury:

- `f_trip_participants`
  - zadaniem funkcji jest zwrócenie listy uczestników wskazanej wycieczki
  - parametry funkcji: `trip_id`
  - funkcja zwraca podobny zestaw danych jak widok `vw_eservation`
- `f_person_reservations`
  - zadaniem funkcji jest zwrócenie listy rezerwacji danej osoby
  - parametry funkcji: `person_id`
  - funkcja zwraca podobny zestaw danych jak widok `vw_reservation`
- `f_available_trips_to`
  - zadaniem funkcji jest zwrócenie listy wycieczek do wskazanego kraju, dostępnych w zadanym okresie czasu (od `date_from` do `date_to`)
  - parametry funkcji: `country`, `date_from`, `date_to`

Funkcje powinny zwracać tabelę/zbiór wynikowy. Należy rozważyć dodanie kontroli parametrów, (np. jeśli parametrem jest `trip_id` to można sprawdzić czy taka wycieczka istnieje). Podobnie jak w przypadku widoków należy zwrócić uwagę na strukturę kodu

Czy kontrola parametrów w przypadku funkcji ma sens?

- jakie są zalety/wady takiego rozwiązania?

Proponowany zestaw funkcji można rozbudować wedle uznania/potrzeb

- np. można dodać nowe/pomocnicze funkcje/procedury

Zadanie 2 - rozwiązanie

```
-- funkcja 1

create function f_trip_participants(trip_id int)
```

```
    return trip_participant_table
as
    result       trip_participant_table;
    v_trip_count int;
begin

    SELECT COUNT(*)
    INTO v_trip_count
    FROM TRIP t
    WHERE t.TRIP_ID = f_trip_participants.trip_id;

    if v_trip_count = 0 then
        raise_application_error(-20001, 'Invalid trip ID');
    end if;
    select trip_participant(res.trip_name, res.firstname, res.lastname,
res.trip_id, res.person_id) bulk collect
    into result
    from vw_reservation res
    where res.trip_id = f_trip_participants.trip_id;
    if result.COUNT = 0 then
        raise_application_error(-20001, 'No participants found for this trip');
    end if;
    return result;
end;

-- funkcja 2

create function f_person_reservations(person_id int)
    return person_reservation_table
as
    result person_reservation_table;
    person_exists int;
begin
    select case
               when exists(    select *
                               from PERSON t
                               where t.PERSON_ID=f_person_reservations.person_id)
then 1
               else 0
               end
    into person_exists from dual;
    if  person_exists=0 then
        raise_application_error(-20001, 'invalid person id');
    end if;
    select person_reservation(res.status,res.firstname, res.lastname,
res.person_id,res.reservation_id,res.trip_id)
            bulk collect
    into result
    from vw_reservation res
    where res.person_id = f_person_reservations.person_id;

    if result.COUNT =0 then
        raise_application_error(-20001, 'No reservations found for this person');
    end if;
```

```
    return result;
end;

--funkcja 3

create function f_available_trips_to(country varchar, date_from date, date_to
date)
    return available_trips_to_table
as
    result available_trips_to_table;
begin

    select available_trips_to(t.TRIP_NAME,t.TRIP_DATE, t.TRIP_ID)
            bulk collect
    into result
    from TRIP t
    where (t.COUNTRY = f_available_trips_to.country and
t.TRIP_DATE>=f_available_trips_to.date_from and
t.TRIP_DATE<=f_available_trips_to.date_to);

    if result.COUNT =0 then
        raise_application_error(-20001, 'No available found for this country in
this period');
    end if;

    return result;
end;
```

Zadanie 3 - procedury

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

Procedury

- `p_add_reservation`
  - zadaniem procedury jest dopisanie nowej rezerwacji
  - parametry: `trip_id`, `person_id`, `no_tickets`
  - procedura powinna kontrolować czy wycieczka jeszcze się nie odbyła, i czy sa wolne miejsca
  - procedura powinna również dopisywać inf. do tabeli `log`
- `p_modify_reservation_status
  - zadaniem procedury jest zmiana statusu rezerwacji
  - parametry: `reservation_id`, `status`

    - procedura powinna 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)
    - procedura powinna również dopisywać inf. do tabeli `log`
- `p_modify_reservation
    - zadaniem procedury jest zmiana statusu rezerwacji
    - parametry: `reservation_id`, `no_iickets`
    - procedura powinna kontrolować czy możliwa jest zmiana liczby sprzedanych/zarezerwowanych biletów – może już nie być miejsc
    - procedura powinna również dopisywać inf. do tabeli `log`
- `p_modify_max_no_places`
    - zadaniem procedury jest zmiana maksymalnej liczby miejsc na daną wycieczkę
    - parametry: `trip_id`, `max_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..)

Proponowany zestaw procedur można rozbudować wedle uznania/potrzeb

- np. można dodać nowe/pomocnicze funkcje/procedury

Zadanie 3 - rozwiązanie

```
--sprawdzenie czy istnieje wycieczka którą chcemy zmodyfikować
create or replace procedure p_check_trip_exists(trip_id in TRIP.TRIP_ID%type)
    as
        tmp char(1);
    begin
        select 1 into tmp from TRIP T where T.TRIP_ID=trip_id;
    exception
        when NO_DATA_FOUND then
            raise_application_error(-20001, 'reservation not found !!!');
    end;

--sprawdzenie czy istnieje rezerwacja którą chcemy zmodyfikować
create or replace procedure p_check_reservation_exists(res_id in
RESERVATION.RESERVATION_ID%type)
    as
        tmp char(1);
    begin
        select 1 into tmp from RESERVATION R where R.RESERVATION_ID=res_id;
    exception
```

```
        when NO_DATA_FOUND then
            raise_application_error(-20001, 'reservation not found !!!');
    end;
--sprawdzenie czy istnieje osoba z podanym id
create procedure p_check_person_exists(per_id in PERSON.PERSON_ID%type)
as
    tmp char(1);
begin
    select 1 into tmp from PERSON P where P.PERSON_ID=per_id;
exception
    when NO_DATA_FOUND then
        raise_application_error(-20001, 'person not found !!!');
end;
-- dodanie rezerwacji i dodanie wpisu do loga, przy weryfikacji argumentow
wejściowych oraz czy są jeszcze  miejsca wolne na wycieczkę
create procedure p_add_reservation(trip_id int, person_id int,
                                   no_tickets int)
as
    curr_date date;
    exist int;
    res_id int;
begin
    curr_date := trunc(sysdate);

    P_CHECK_PERSON_EXISTS(person_id);

    P_CHECK_RESERVATION_EXISTS(trip_id);

    begin
        select 1 into exist
                from VW_AVAILABLE_TRIP t
                where p_add_reservation.trip_id =t.TRIP_ID and
                t.TRIP_DATE> curr_date and
                p_add_reservation.no_tickets<= t.NO_AVAILABLE_PLACES;
    exception
            when NO_DATA_FOUND then
                    raise_application_error(-20001, 'No available places for
reservation');
    end;
    insert into RESERVATION(TRIP_ID,PERSON_ID,STATUS,NO_TICKETS)
    values(
           p_add_reservation.trip_id,
           p_add_reservation.person_id,
           'N',
           p_add_reservation.no_tickets)
    returning reservation_id into res_id;
    insert into LOG(RESERVATION_ID,LOG_DATE,STATUS,NO_TICKETS)
    values(
              res_id,
              curr_date,
              'N',
              p_add_reservation.no_tickets);
end;
-- modyfkiacja statusu rezerwacji i dodanie wpisu do loga, przy zmianie z
```

```
anulowanej na aktywna sprawdzenie czy to możliwe
create procedure p_modify_reservation_status (
    v_reservation_id number,
    v_status char
) as
    curr_date date;
    no_tickets number := 0;
    curr_res_status char(1);
begin
    curr_date := trunc(sysdate);
    p_check_reservation_exists(v_reservation_id);

    select r.status into curr_res_status
    from reservation r
    where r.reservation_id = v_reservation_id;

    if curr_res_status = 'C' then
        begin
            select no_tickets into no_tickets
            from vw_reservation t
            where v_reservation_id = t.reservation_id
              and t.no_tickets <= f_get_available_places(t.RESERVATION_ID);
        exception
            when no_data_found then
                raise_application_error(-20001, 'No available places for
reservation');
        end;
    end if;

    update reservation
    set status = v_status
    where reservation_id = v_reservation_id;

    insert into log (reservation_id, log_date, status, no_tickets)
    values (
               v_reservation_id,
               curr_date,
               v_status,
               no_tickets
           );
end;


-- modyfikacja rezerwacji (liczby biletów, nie może ona przekroczyć dostępnej
liczby), zapisanie do LOG
create or replace procedure p_modify_reservation(res_id in
RESERVATION.RESERVATION_ID%type, no_tickets number)
    as
        trip_id TRIP.TRIP_ID%type;
        available_places number;
        current_no_tickets number;
        res_status varchar2(10);
    begin
        p_check_reservation_exists(res_id);
```

```
        select R.TRIP_ID, R.NO_TICKETS, R.STATUS
        into trip_id, current_no_tickets, res_status
        from RESERVATION R
        where R.RESERVATION_ID = res_id;

        available_places:=F_GET_AVAILABLE_PLACES(trip_id)

        if (available_places-(no_tickets-current_no_tickets))<0 then
            raise_application_error(-20001, 'not enough free places');
        end if;

        update RESERVATION R
            set NO_TICKETS=no_tickets
        where R.RESERVATION_ID=res_id;

        insert into LOG(reservation_id, log_date, status, no_tickets)
values(res_id, CURRENT_DATE, res_status, no_tickets);
    end;

-- modyfikacja wycieczki (maksymalnej liczby miejsc, nie może ona być mniejsza od
zarezerwowanych już miejsc)
create procedure p_modify_max_no_places(trip_id in TRIP.TRIP_ID%type,
max_no_places number)
    as
        booked_places number;
    begin
        p_check_trip_exists(trip_id);

        select coalesce(sum(NO_TICKETS), 0) into booked_places from RESERVATION R
where R.TRIP_ID=p_modify_max_no_places.trip_id;

        if booked_places>max_no_places then
            raise_application_error(-20001, 'max_no_places lower than no of booked
places');
        end if;

        update TRIP T
            set T.MAX_NO_PLACES=p_modify_max_no_places.max_no_places
        where T.TRIP_ID=p_modify_max_no_places.trip_id;
    end;
```

Zadanie 4 - triggery

Zmiana strategii zapisywania do dziennika rezerwacji. Realizacja przy pomocy triggerów

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

Triggery:

- trigger/triggery obsługujące
  - dodanie rezerwacji
  - zmianę statusu
  - zmianę liczby zarezerwowanych/kupionych biletów
- trigger zabraniający usunięcia rezerwacji

Oczywiście po wprowadzeniu tej zmiany należy "uaktualnić" procedury modyfikujące dane.

> UWAGA Należy stworzyć nowe wersje tych procedur (dodając do nazwy dopisek 4 - od numeru zadania). Poprzednie wersje procedur należy pozostawić w celu umożliwienia weryfikacji ich poprawności

Należy przygotować procedury: `p_add_reservation_4`, `p_modify_reservation_status_4` , `p_modify_reservation_4`

Zadanie 4 - rozwiązanie

```
-- wpisanie do log po aktualizacji lub nowym wpisie do RESERVATION
create or replace trigger tr_log_ins_upd
    after insert or update
    on RESERVATION
    for each row
        begin
            insert into LOG(reservation_id, log_date, status, no_tickets)
            values (:NEW.RESERVATION_ID, CURRENT_DATE, :NEW.STATUS,
:NEW.NO_TICKETS);
        end;

-- blokada usunięcia wpisu z RESERVATION
create or replace trigger tr_reservation_del
    before delete
    on RESERVATION
    begin
        raise_application_error(-20001,'reservations can not be deleted');
    end;
-- taka sama funkcja jak porzednio, bez logów
create procedure p_add_reservation_4(
    trip_id int, person_id int, no_tickets int)
as
    exist int;
begin

    P_CHECK_PERSON_EXISTS(person_id);

    P_CHECK_RESERVATION_EXISTS(trip_id);
    begin
```

```
        select 1 into exist
        from VW_AVAILABLE_TRIP t
        where p_add_reservation_4.trip_id =t.TRIP_ID and
            t.TRIP_DATE> CURRENT_DATE and
            p_add_reservation_4.no_tickets<= t.NO_AVAILABLE_PLACES;
    exception
        when NO_DATA_FOUND then
            raise_application_error(-20001, 'No available places for
reservation');
    end;
    insert into RESERVATION(TRIP_ID,PERSON_ID,STATUS,NO_TICKETS)
    values(
              p_add_reservation_4.trip_id,
              p_add_reservation_4.person_id,
              'N',
              p_add_reservation_4.no_tickets);
end;
-- taka sama funkcja jak porzednio, bez logów
create procedure p_modify_reservation_status_4 (
    v_reservation_id number,
    v_status char
) as
    no_tickets number := 0;
    curr_res_status char(1);
begin
    p_check_reservation_exists(v_reservation_id);

    select r.status into curr_res_status
    from reservation r
    where r.reservation_id = v_reservation_id;

    if curr_res_status = 'C' then
        begin
            select no_tickets into no_tickets
            from vw_reservation t
            where v_reservation_id = t.reservation_id
              and t.no_tickets <= f_get_available_places(t.RESERVATION_ID);
        exception
            when no_data_found then
                raise_application_error(-20001, 'No available places for
reservation');
        end;
    end if;

    update reservation
    set status = v_status
    where reservation_id = v_reservation_id;
end;
-- taka sama funkcja jak porzednio, bez logów
create procedure p_modify_reservation_4(res_id in RESERVATION.RESERVATION_ID%type,
no_tickets number)
as
    trip_id TRIP.TRIP_ID%type;
    available_places number;
```

```
    current_no_tickets number;
    res_status varchar2(10);
begin
    p_check_reservation_exists(res_id);

    select R.TRIP_ID, R.NO_TICKETS, R.STATUS
    into trip_id, current_no_tickets, res_status
    from RESERVATION R
    where R.RESERVATION_ID = res_id;

    available_places:=F_GET_AVAILABLE_PLACES(trip_id);

    if (available_places-(no_tickets-current_no_tickets))<0 then
        raise_application_error(-20001, 'not enough free places');
    end if;
    if (res_status<>'N') then
        raise_application_error(-20001, 'wrong application status');
    end if;

    update RESERVATION R
    set R.NO_TICKETS=p_modify_reservation_4.no_tickets
    where R.RESERVATION_ID=res_id;

end;
```

Zadanie 5 - triggery

Zmiana strategii kontroli dostępności miejsc. Realizacja przy pomocy triggerów

Należy wprowadzić zmianę, która spowoduje, że kontrola dostępności miejsc na wycieczki (przy dodawaniu nowej rezerwacji, zmianie statusu) będzie realizowana przy pomocy trigerów

Triggery:

- Trigger/triggery obsługujące:
  - dodanie rezerwacji
  - zmianę statusu
  - zmianę liczby zakupionych/zarezerwowanych miejsc/biletów

Oczywiście po wprowadzeniu tej zmiany należy "uaktualnić" procedury modyfikujące dane.

> UWAGA Należy stworzyć nowe wersje tych procedur (np. dodając do nazwy dopisek 5 - od numeru zadania). Poprzednie wersje procedur należy pozostawić w celu umożliwienia weryfikacji ich poprawności.

Należy przygotować procedury: `p_add_reservation_5`, `p_modify_reservation_status_5`, `p_modify_reservation_status_5`

Zadanie 5 - rozwiązanie

```
-- kontrola czy po zmianie statusu istniejącej rezerwacji lub dodaniu nowej nie
zostaje przekroczona liczba wolnych miejsc
-- compund trigger uzyty po to by nie musiec sie odwolywac do jedcześnie
modyfikowanej tabeli
create trigger TR_RESERVATION_INS_UPD
    instead of insert or update
    on RESERVATION
    for each row
    compound trigger

    type reservation_rec is record (
                                        reservation_id
reservation.reservation_id%type,
                                        trip_id reservation.trip_id%type,
                                        status reservation.status%type,
                                        no_tickets reservation.no_tickets%type
                                    );

    type reservation_tab is table of reservation_rec index by binary_integer;
    reservations reservation_tab;
    ind number := 0;

before each row is
begin
    ind := ind + 1;
    reservations(ind).reservation_id := :new.reservation_id;
    reservations(ind).trip_id := :new.trip_id;
    reservations(ind).status := :new.status;
    reservations(ind).no_tickets := :new.no_tickets;
end before each row;

    after statement is
    begin
        for i in 1..ind loop
                if reservations(i).status <> 'C' then
                    declare
                        available_places number;
                    begin
                        select f_get_available_places(reservations(i).trip_id)
                        into available_places
                        from dual;

                        if reservations(i).no_tickets > available_places then
                            raise_application_error(-20001, 'not enough available
places');
                        end if;
```

```
                    end;
                end if;
            end loop;
    end after statement;
    end tr_reservation_ins_upd;
-- procedura już bez weryfikacji liczby miejsc
create or replace procedure p_add_reservation_5(trip_id int, person_id int,
                                    no_tickets int)
as
    curr_date date;
begin
    curr_date := trunc(sysdate);

    P_CHECK_PERSON_EXISTS(person_id);

    P_CHECK_RESERVATION_EXISTS(trip_id);

    insert into RESERVATION(TRIP_ID,PERSON_ID,STATUS,NO_TICKETS)
    values(
              p_add_reservation_5.trip_id,
              p_add_reservation_5.person_id,
              'N',
              p_add_reservation_5.no_tickets);
end;
-- procedura już bez weryfikacji liczby miejsc
create or replace procedure p_modify_reservation_5(res_id in
RESERVATION.RESERVATION_ID%type, pno_tickets number)
    as
        res_status char(1);
    begin
        p_check_reservation_exists(res_id);

        select R.TRIP_ID, R.NO_TICKETS, R.STATUS into res_status
        from RESERVATION R
        where R.RESERVATION_ID = res_id;

        update RESERVATION R
            set NO_TICKETS=pno_tickets
        where R.RESERVATION_ID=res_id;
    end;
-- procedura już bez weryfikacji liczby miejsc
create or replace procedure p_modify_reservation_status_5 (
    v_reservation_id number,
    v_status char
) as
    curr_res_status char(1);
begin
    p_check_reservation_exists(v_reservation_id);

    select r.status
    into curr_res_status
    from reservation r
    where r.reservation_id = v_reservation_id;
```

```
    update reservation
    set status = v_status
    where reservation_id = v_reservation_id;
end;
```

Zadanie 6

Zmiana struktury bazy danych. W tabeli `trip` należy dodać redundantne pole `no_available_places`. Dodanie redundantnego pola uprości kontrolę dostępnych miejsc, ale nieco skomplikuje procedury dodawania rezerwacji, zmiany statusu czy też zmiany maksymalnej liczby miejsc na wycieczki.

Należy przygotować polecenie/procedurę przeliczającą wartość pola `no_available_places` dla wszystkich wycieczek (do jednorazowego wykonania)

Obsługę pola `no_available_places` można zrealizować przy pomocy procedur lub triggerów

Należy zwrócić uwagę na spójność rozwiązania.

> UWAGA Należy stworzyć nowe wersje tych widoków/procedur/triggerów (np. dodając do nazwy dopisek 6 - od numeru zadania). Poprzednie wersje procedur należy pozostawić w celu umożliwienia weryfikacji ich poprawności.

- zmiana struktury tabeli

```
alter table trip add
    no_available_places int null
```

- polecenie przeliczające wartość `no_available_places`
  - należy wykonać operację "przeliczenia" liczby wolnych miejsc i aktualizacji pola `no_available_places`

Zadanie 6 - rozwiązanie

```
-- wpisanie aktualnej liczby miejsc do TRIP.NO_AVAILABLE_PLACES
create or replace procedure p_trip_update_no_available_places is
    begin
    update TRIP t
    set t.NO_AVAILABLE_PLACES = t.MAX_NO_PLACES - (
        select coalesce(sum(r.no_tickets), 0)
        from RESERVATION r
        where r.TRIP_ID = t.TRIP_ID AND r.STATUS <> 'C'
```

```
    );
    end;


-- zmieniony trigger który korzysta z pola NO_AVAILABLE_PLACES zamiast z funkcji
obliczającej wolne miejsca
create trigger TR_RESERVATION_INS_UPD_6
    instead of insert or update
    on RESERVATION
    for each row
    compound trigger

    type reservation_rec is record (
                                        reservation_id
reservation.reservation_id%type,
                                        trip_id reservation.trip_id%type,
                                        status reservation.status%type,
                                        no_tickets reservation.no_tickets%type
                                    );

    type reservation_tab is table of reservation_rec index by binary_integer;
    reservations reservation_tab;
    ind number := 0;

before each row is
begin
    ind := ind + 1;
    reservations(ind).reservation_id := :new.reservation_id;
    reservations(ind).trip_id := :new.trip_id;
    reservations(ind).status := :new.status;
    reservations(ind).no_tickets := :new.no_tickets;
end before each row;

    after statement is
    begin
        for i in 1..ind loop
                if reservations(i).status <> 'C' then
                    declare
                        available_places number;
                    begin
                        select no_available_places
                        into available_places
                        from trip t
                        where t.id = reservations(i).trip_id;

                        if reservations(i).no_tickets > available_places then
                            raise_application_error(-20001, 'not enough available
places');
                        end if;
                    end;
                end if;
            end loop;
    end after statement;
    end tr_reservation_ins_upd;
```

Zadanie 6a - procedury

Obsługę pola `no_available_places` należy zrealizować przy pomocy procedur

- procedura dodająca rezerwację powinna aktualizować pole `no_available_places` w tabeli trip
- podobnie procedury odpowiedzialne za zmianę statusu oraz zmianę maksymalnej liczby miejsc na wycieczkę
- należy przygotować procedury oraz jeśli jest to potrzebne, zaktualizować triggery oraz widoki

> UWAGA Należy stworzyć nowe wersje tych widoków/procedur/triggerów (np. dodając do nazwy dopisek 6a - od numeru zadania). Poprzednie wersje procedur należy pozostawić w celu umożliwienia weryfikacji ich poprawności.

- może być potrzebne wyłączenie 'poprzednich wersji' triggerów

Zadanie 6a - rozwiązanie

```
-- zmieniona procedura aktualizująca liczbę wolnych miejsc po zmianie maksymalnej
liczby miejsc na wycieczkę
create or replace procedure p_modify_max_no_places_6a(trip_id in
TRIP.TRIP_ID%type, max_no_places number)
    as
        cur_max_places number;
        cur_av_places number;
    begin
        p_check_trip_exists(trip_id);

        select MAX_NO_PLACES, NO_AVAILABLE_PLACES into cur_max_places,
cur_av_places from TRIP T where T.TRIP_ID=trip_id;

        if max_no_places-cur_max_places>cur_av_places then
            raise_application_error(-20001, 'max_no_places dif lower than no of
available places');
        end if;

        update TRIP T
            set T.MAX_NO_PLACES=max_no_places, T.NO_AVAILABLE_PLACES =
cur_av_places - (max_no_places - cur_max_places)
        where T.TRIP_ID=trip_id;
    end;


-- zmieniona procedura aktualizująca liczbę wolnych miejsc na wycieczce po
```

```
modyfikacji liczby biletów w rezerwacji
create or replace procedure p_modify_reservation_6a(res_id in
RESERVATION.RESERVATION_ID%type, pno_tickets number)
as
    res_status varchar2(10);
    cur_no_tickets number;
    cur_trip_id TRIP.TRIP_ID%type;
begin
    p_check_reservation_exists(res_id);

    select R.TRIP_ID, R.NO_TICKETS, R.STATUS into
cur_trip_id,cur_no_tickets,res_status
    from RESERVATION R
    where R.RESERVATION_ID = res_id;

    update RESERVATION R
        set NO_TICKETS=pno_tickets
    where R.RESERVATION_ID=res_id;

    update TRIP T
        set T.NO_AVAILABLE_PLACES = T.NO_AVAILABLE_PLACES - (pno_tickets -
cur_no_tickets)
    where T.TRIP_ID=cur_trip_id;
end;

-- -- zmieniona procedura aktualizująca liczbę wolnych miejsc na wycieczce po
modyfikacji statusu rezerwacji
create procedure p_modify_reservation_status_6a (
    v_reservation_id number,
    v_status char
) as
    curr_date date;
    no_tickets number := 0;
    curr_res_status char(1);
begin
    curr_date := trunc(sysdate);
    p_check_reservation_exists(v_reservation_id);

    select r.status into curr_res_status
    from reservation r
    where r.reservation_id = v_reservation_id;

    if curr_res_status = 'C' then
        begin
            select no_tickets into no_tickets
            from vw_reservation t
            where v_reservation_id = t.reservation_id
              and t.no_tickets <= f_get_available_places(t.RESERVATION_ID);
        exception
            when no_data_found then
                raise_application_error(-20001, 'No available places for
reservation');
        end;
    end if;
```

```
    update reservation
    set status = v_status
    where reservation_id = v_reservation_id;

    insert into log (reservation_id, log_date, status, no_tickets)
    values (
               v_reservation_id,
               curr_date,
               v_status,
               no_tickets
           );

    if curr_res_status = 'C' and v_status <> 'C' then
        update trip
           set no_available_places = no_available_places - no_tickets
         where trip_id = v_trip_id;
    elsif curr_res_status <> 'C' amd v_status = 'C' then
        update trip
           set no_available_places = no_available_places + no_tickets
         where trip_id = v_trip_id;
    end if;
end;
```

Zadanie 6b - triggery

Obsługę pola `no_available_places` należy zrealizować przy pomocy triggerów

- podczas dodawania rezerwacji trigger powinien aktualizować pole `no_available_places` w tabeli trip
- podobnie, podczas zmiany statusu rezerwacji
- należy przygotować trigger/triggery oraz jeśli jest to potrzebne, zaktualizować procedury modyfikujące dane oraz widoki

> UWAGA Należy stworzyć nowe wersje tych widoków/procedur/triggerów (np. dodając do nazwy dopisek 6b - od numeru zadania). Poprzednie wersje procedur należy pozostawić w celu umożliwienia weryfikacji ich poprawności.

- może być potrzebne wyłączenie 'poprzednich wersji' triggerów

Zadanie 6b - rozwiązanie

```
--trigger aktualizujący liczbę wolnych miejsc na wycieczce w zależności od tego
czy zmienił się status rezerwacji, czy liczba biletów
create or replace trigger tr_update_available_places
    after update
```

```
    on RESERVATION
    for each row
    begin
        if :NEW.STATUS<>'C' and :OLD.STATUS='C' then
            UPDATE trip t
            SET t.no_available_places = t.NO_AVAILABLE_PLACES - :NEW.NO_TICKETS
            WHERE t.TRIP_ID = :NEW.TRIP_ID;
        end if;
        if :NEW.STATUS='C' and :OLD.STATUS<>'C' then
            UPDATE trip t
            SET t.no_available_places = t.NO_AVAILABLE_PLACES + :NEW.NO_TICKETS
            WHERE t.TRIP_ID = :NEW.TRIP_ID;
        end if;
        if :NEW.STATUS=:OLD.STATUS and :OLD.STATUS<>'C' and
:NEW.NO_TICKETS<>:OLD.NO_TICKETS then
            UPDATE trip t
            SET t.no_available_places = t.NO_AVAILABLE_PLACES - (:NEW.NO_TICKETS -
:OLD.NO_TICKETS)
            WHERE t.TRIP_ID = :NEW.TRIP_ID;
        end if;
    end;


--trigger aktualizujący liczbę wolnych miejsc na wycieczce po dodaniu nowej
rezerwacji
create or replace trigger tr_insert_available_places
    after insert
    on RESERVATION
    for each row
    begin
        if :NEW.STATUS<>'C' then
            UPDATE trip t
            SET t.no_available_places = t.NO_AVAILABLE_PLACES - :NEW.NO_TICKETS
            WHERE t.TRIP_ID = :NEW.TRIP_ID;
        end if;
    end;
```

Modyfikacja procedur polega na cofnięciu zmian wprowadzonych w podpunkcie 6a.

Zadanie 7 - podsumowanie

Porównaj sposób programowania w systemie Oracle PL/SQL ze znanym ci systemem/językiem MS Sqlserver T-SQL

Różnice występują między innymi w:
-deklaracji funkcji i procedur (argumenty oraz zadeklarowane zmienne). Bardziej przypadła nam do gustu forma PL/SQL, -w transakcjach, nie można ich zagnieżdżać,

-w typach danych,
-w obsłudze wyjątków