Oracle PL/Sql

widoki, funkcje, procedury, triggery ćwiczenie

Imiona i nazwiska autorów : Wojciech Wietrzny, Antoni Dulewicz Marcin Serafin

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
  - `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 datetime  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');
```

Zadanie 0 - modyfikacja danych, transakcje

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

Skomentuj działanie 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=214774 w szczególności dokument: `1_modyf.pdf`

```
begin
    update trip set country = 'Niemcy' where trip_id = 1;
    rollback;
end;

begin
    insert into person(firstname, lastname) values ('Żaneta', 'Kowalska');
    insert into person(firstname, lastname) values ('Janek', 'Rapowanie');
    commit;
end;
begin
    update person set person_id = 11 where firstname = 'Żaneta' and lastname =
'Kowalska';
    commit;
end;
begin
    update person set person_id = 12 where firstname = 'Janek' and lastname =
'Rapowanie';
    commit;
end;
-- bez commita nie usuwa z tabeli
begin
    delete from person where person_id = 11;
    delete from person where person_id = 12;
end;


begin
    delete from person where person_id = 11;
    delete from person where person_id = 12;
```

```
    commit;
end;


Transakcjami nazywami zagnieżdżone w blokach begin end linijki kodu,
wykonujące pewne operacje na systemie bazy danych. Do transakcji używane są
polecenia
commit i rollback - rollback wycofuje wprowadzenie zmian, commit natomiast
zatwierdza
zmiany do bazy danych. W praktyce polecenia rollback używamy zabezpieczając się
przed
potencjalnymi błędami, tak żeby zmiany nie zaszły częściowo, jeśli wprowadzone
dane
nie są poprawne bądź występują jakieś inne problemy skutkujące wyrzuceniem błędu -
usuwanie
części danych ręcznie, zeby nastepnie wprowadzic holistycznie calosc zmian byloby
niepraktyczne.
Języki MS SQLserver T-SQL i Oracle PL/SQL mają wiele podobieństw w sposobie
wykorzystywania
transakcji, w obu mozemy dane w transakcjach wstawiac, usuwac. Oprocz tego uzywac
polecen commit i rollback, definiowac bloki transakcji. Z istotniejszych roznic
mozna wskazac obsluge bledow - w PL/SQL wystepuja exceptions zamiast blokow try
catch w T-SQLu.
Dodatkowo transakcje w PL/SQL są obsługiwane w sposób niejawny - każde polecenie
SQL jest
domyślnie częścią transakcji, a w T-SQL mamy to zrobione w sposób jawny
```

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`
- `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
- np. można zmienić def. widoków, dodając nowe/potrzebne pola

Zadanie 1 - rozwiązanie

```
-- Widok rezerwacji
create view vw_reservation as
select
    r.RESERVATION_ID,
    t.COUNTRY,
    t.TRIP_DATE,
    t.TRIP_NAME,
    p.FIRSTNAME,
    p.LASTNAME,
    r.STATUS,
    t.TRIP_ID,
    p.PERSON_ID
from
    RESERVATION r
join
        trip t on r.TRIP_ID = t.TRIP_ID
join
        person p on r.PERSON_ID = p.PERSON_ID
order by r.RESERVATION_ID;


-- Widok który dla każdej wycieczki podaje liczbę już zajętych miejsc
create view vw_taken_places as
select distinct
    r.TRIP_ID,
    (select count(*) from RESERVATION r1
    where r.TRIP_ID=r1.TRIP_ID and (r1.STATUS !='C')
    group by r1.TRIP_ID) as taken_places
from RESERVATION r;

-- Widok wszystkich wycieczek z użyciem widoku vw_taken_places
create view vw_trip as
select
    t.TRIP_ID,
    t.COUNTRY,
    t.TRIP_DATE,
    t.TRIP_NAME,
    t.MAX_NO_PLACES,
    t.MAX_NO_PLACES - (select t1.TAKEN_PLACES
     from VW_TAKEN_PLACES t1
     where t1.TRIP_ID = t.TRIP_ID) as no_available_places
from TRIP t
```

```
-- Widok tylko dostępnych wycieczek
create view vw_available_trip as
select
    *
from VW_TRIP
where TRIP_DATE > SYSDATE and NO_AVAILABLE_PLACES > 0;
```

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 pomocnicza do sprawdzania czy istnieje trip o danym id
create function f_trip_exists(t_id in TRIP.TRIP_ID%type)
    return boolean
as
    exist number;
begin
    select case
            when exists(select * from TRIP where TRIP_ID = t_id) then 1
            else 0
        end
    into exist from dual;

    if exist = 1 then
        return true;
    else
        return false;
    end if;
end;
/



--f_trip_participants
create function f_trip_participants(t_id int)
    return TRIP_PARTICIPANTS_TABLE
as
    result TRIP_PARTICIPANTS_TABLE;
    vaild int;
begin

    if not F_TRIP_EXISTS(t_id) then
        raise_application_error(-20001,'trip not found');
    end if;

    select
TRIP_PARTICIPANT(PERSON_ID,FIRSTNAME,LASTNAME,RESERVATION_ID,COUNTRY,TRIP_DATE,TRI
P_NAME,STATUS)
    bulk collect
    into result
    from VW_RESERVATION
    where TRIP_ID = t_id and STATUS != 'C';

    return result;
end;
/



-- funkcja pomocnicza sprawdzająca czy osoba o danym id istnieje
create function f_person_exists(p_id in person.PERSON_ID%type)
    return boolean
as
    exist number;
```

```
begin
    select case
            when exists(select * from PERSON where PERSON_ID = p_id) then 1
            else 0
        end
    into exist from dual;

    if exist = 1 then
        return true;
    else
        return false;
    end if;
end;
/




--f_person_reservations
create or replace function f_person_reservations(p_id int )
    return person_reservation_table
as
    result person_reservation_table;
    valid int;
begin

    if not F_PERSON_EXISTS(p_id) then
        raise_application_error(-20001,'person not found');
    end if;

    select
person_reservation(PERSON_ID,RESERVATION_ID,TRIP_ID,FIRSTNAME,LASTNAME,COUNTRY,TRI
P_DATE,TRIP_NAME,STATUS)
    bulk collect
    into result
    from VW_RESERVATION
    where PERSON_ID = p_id;

    return result;
end;



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

    select
available_trip_to(TRIP_ID,COUNTRY,TRIP_DATE,TRIP_NAME,MAX_NO_PLACES,NO_AVAILABLE_P
LACES)
```

```
    bulk collect
    into result
    from VW_AVAILABLE_TRIP
    where COUNTRY = f_country and TRIP_DATE between date_from and date_to;

    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`,
  - 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_tatus`
  - 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`

Procedury:

- `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

```
-- Dodawanie Rezerwacji
create or replace procedure p_add_reservation(
    p_trip_id in NUMBER,
    p_person_id in Number
) AS
    v_trip_date DATE;
    v_available_seats NUMBER;
    v_reservation_id NUMBER;
    v_log_id NUMBER;
    v_trip_number NUMBER;
BEGIN
    select count(*) into v_trip_number from TRIP where TRIP_ID=p_trip_id;
    if v_trip_number>0 THEN
        select TRIP_DATE into v_trip_date
        from TRIP
            where TRIP_ID=p_trip_id;
        IF v_trip_date>SYSDATE then
            Select MAX_NO_PLACES - (select count(*) from RESERVATION where
TRIP.TRIP_ID=RESERVATION.TRIP_ID and RESERVATION.STATUS!='C')
                into v_available_seats
            from TRIP
                where TRIP_ID=p_trip_id;
            IF v_available_seats>0 then
                select Max(RESERVATION_ID)+1 into v_reservation_id from
RESERVATION;
                select Max(LOG_ID)+1 into v_log_id from LOG;
                Insert into RESERVATION (reservation_id, trip_id, person_id,
status) VALUES
                (v_reservation_id,p_trip_id,p_person_id,'P');
                INSERT INTO log (LOG_ID, RESERVATION_ID, LOG_DATE, STATUS)
                VALUES (v_log_id, v_reservation_id,SYSDATE,'P');
            ELSE
                DBMS_OUTPUT.PUT_LINE('No available seats for this trip.');
            end if;
            ELSE
                DBMS_OUTPUT.PUT_LINE('Trip has already been completed.');
        end if;
    ELSE
        DBMS_OUTPUT.PUT_LINE('There is no Trip with that TRIP_ID');
    end if;
end;
/

-- Zmiana Rezerwacji
```

```
CREATE OR REPLACE PROCEDURE p_modify_reservation_status (
    p_reservation_id IN NUMBER,
    p_status IN VARCHAR2
) AS
    v_current_status VARCHAR2(20);
    v_trip_id NUMBER;
    v_trip_date DATE;
    v_log_id NUMBER;
    v_available_seats NUMBER;
    v_reservation_number NUMBER;
BEGIN
    select count(*) into v_reservation_number from RESERVATION where
RESERVATION_ID=p_reservation_id;
    IF v_reservation_number>0 THEN
        SELECT status INTO v_current_status
        FROM RESERVATION
        WHERE reservation_id = p_reservation_id;
        IF v_current_status != p_status THEN
            select TRIP_ID into v_trip_id from RESERVATION where
RESERVATION_ID=p_reservation_id;
            select TRIP_DATE into v_trip_date from TRIP where TRIP_ID=v_trip_id;
            IF v_trip_date>SYSDATE THEN
                Select MAX_NO_PLACES - (select count(*) from RESERVATION where
TRIP.TRIP_ID=RESERVATION.TRIP_ID and RESERVATION.STATUS!='C')
                into v_available_seats
                from TRIP
                    where TRIP_ID=v_trip_id;
                IF v_available_seats>0 THEN
                    UPDATE RESERVATION
                    SET status = p_status
                    WHERE reservation_id = p_reservation_id;
                    select Max(LOG_ID)+1 into v_log_id from LOG;
                    INSERT INTO log (LOG_ID, RESERVATION_ID, LOG_DATE, STATUS)
                    VALUES (v_log_id, p_reservation_id,SYSDATE,p_status);
                ELSE
                    DBMS_OUTPUT.PUT_LINE('Cannot modify status of a reservation
for full trip.');
                END IF;
            ELSE
            DBMS_OUTPUT.PUT_LINE('Cannot modify status of a reservation for ended
trip.');
            END IF;
        ELSE
            DBMS_OUTPUT.PUT_LINE('Cannot modify status for same status.');
        END IF;
    ELSE
            DBMS_OUTPUT.PUT_LINE('There is no Reservation with that
RESERVATION_ID');
    END IF;
END;
/

--Zmiana liczby miejsc na wyjazd
CREATE OR REPLACE PROCEDURE p_modify_max_no_places (
```

```
    p_trip_id IN NUMBER,
    p_max_no_places IN NUMBER
) AS
    v_reserved_seats NUMBER;
    v_trip_number NUMBER;
BEGIN
    select count(*) into v_trip_number from TRIP where TRIP_ID=p_trip_id;
    if v_trip_number>0 THEN
        Select (select count(*) from RESERVATION where
TRIP.TRIP_ID=RESERVATION.TRIP_ID)
        into v_reserved_seats
        from TRIP
        where TRIP_ID=p_trip_id;
        IF p_max_no_places >= v_reserved_seats THEN
            UPDATE TRIP
            SET max_no_places = p_max_no_places
            WHERE trip_id = p_trip_id;
        ELSE
            DBMS_OUTPUT.PUT_LINE('Cannot decrease max number of places below
reserved seats.');
        END IF;
    ELSE
        DBMS_OUTPUT.PUT_LINE('There is no Trip with that TRIP_ID.');
    END IF;
END;
```

Zadanie 4 - triggery

Zmiana strategii zapisywania do dziennika rezerwacji. Realizacja przy pomocy triggerów

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

Triggery:

- trigger/triggery obsługujące
  - dodanie rezerwacji
  - zmianę statusu
- 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`

Zadanie 4 - rozwiązanie

```
-- Trigger dodający dodanie Rezerwacji do Logów
CREATE OR REPLACE TRIGGER trg_add_reservation_log
AFTER INSERT ON RESERVATION
FOR EACH ROW
BEGIN
    INSERT INTO LOG (LOG_ID, RESERVATION_ID, LOG_DATE, STATUS)
    VALUES ((SELECT NVL(MAX(log_id), 0) + 1 FROM LOG),
            :NEW.reservation_id,
            SYSDATE,
            'P');
END;
/

-- Trigger dodający zmiane statusu Rezerwacji do Logów
CREATE OR REPLACE TRIGGER trg_modify_reservation_status_log
AFTER UPDATE OF status ON RESERVATION
FOR EACH ROW
BEGIN
    INSERT INTO LOG (LOG_ID, RESERVATION_ID, LOG_DATE, STATUS)
    VALUES ((SELECT NVL(MAX(log_id), 0) + 1 FROM LOG),
            :OLD.reservation_id,
            SYSDATE,
            :NEW.status);
END;
/

-- Trigger blokujący usuwanie Rezerwacji
CREATE OR REPLACE TRIGGER trg_prevent_reservation_deletion
BEFORE DELETE ON RESERVATION
FOR EACH ROW
BEGIN
    RAISE_APPLICATION_ERROR(-20001, 'Deletion of reservation not allowed.');
END;
/

-- Dodawanie Rezerwacji (nowe)
create or replace procedure p_add_reservation_4(
    p_trip_id in NUMBER,
    p_person_id in Number
) AS
    v_trip_date DATE;
    v_available_seats NUMBER;
```

```
    v_reservation_id NUMBER;
    v_log_id NUMBER;
    v_trip_number NUMBER;
BEGIN
    select count(*) into v_trip_number from TRIP where TRIP_ID=p_trip_id;
    if v_trip_number>0 THEN
        select TRIP_DATE into v_trip_date
        from TRIP
            where TRIP_ID=p_trip_id;
        IF v_trip_date>SYSDATE then
            Select MAX_NO_PLACES - (select count(*) from RESERVATION where
TRIP.TRIP_ID=RESERVATION.TRIP_ID and RESERVATION.STATUS!='C')
                into v_available_seats
            from TRIP
                where TRIP_ID=p_trip_id;
            IF v_available_seats>0 then
                select Max(RESERVATION_ID)+1 into v_reservation_id from
RESERVATION;
                select Max(LOG_ID)+1 into v_log_id from LOG;
                Insert into RESERVATION (reservation_id, trip_id, person_id,
status) VALUES
                (v_reservation_id,p_trip_id,p_person_id,'P');
            ELSE
                DBMS_OUTPUT.PUT_LINE('No available seats for this trip.');
            end if;
            ELSE
                DBMS_OUTPUT.PUT_LINE('Trip has already been completed.');
        end if;
    ELSE
        DBMS_OUTPUT.PUT_LINE('There is no Trip with that TRIP_ID');
    end if;
end;
/


/

-- Zmiana statusus Rezerwacji (nowe)
create or replace PROCEDURE p_modify_reservation_status_4 (
    p_reservation_id IN NUMBER,
    p_status IN VARCHAR2
) AS
    v_current_status VARCHAR2(20);
    v_trip_id NUMBER;
    v_trip_date DATE;
    v_log_id NUMBER;
    v_available_seats NUMBER;
    v_reservation_number NUMBER;
BEGIN
    select count(*) into v_reservation_number from RESERVATION where
RESERVATION_ID=p_reservation_id;
    IF v_reservation_number>0 THEN
        SELECT status INTO v_current_status
        FROM RESERVATION
```

```
        WHERE reservation_id = p_reservation_id;
        IF v_current_status != p_status THEN
            select TRIP_ID into v_trip_id from RESERVATION where
RESERVATION_ID=p_reservation_id;
            select TRIP_DATE into v_trip_date from TRIP where TRIP_ID=v_trip_id;
            IF v_trip_date>SYSDATE THEN
                Select MAX_NO_PLACES - (select count(*) from RESERVATION where
TRIP.TRIP_ID=RESERVATION.TRIP_ID and RESERVATION.STATUS!='C')
                into v_available_seats
                from TRIP
                    where TRIP_ID=v_trip_id;
                IF v_available_seats>0 THEN
                    UPDATE RESERVATION
                    SET status = p_status
                    WHERE reservation_id = p_reservation_id;
                    select Max(LOG_ID)+1 into v_log_id from LOG;
                ELSE
                    DBMS_OUTPUT.PUT_LINE('Cannot modify status of a reservation
for full trip.');
                END IF;
            ELSE
            DBMS_OUTPUT.PUT_LINE('Cannot modify status of a reservation for ended
trip.');
            END IF;
        ELSE
            DBMS_OUTPUT.PUT_LINE('Cannot modify status for same status.');
        END IF;
    ELSE
            DBMS_OUTPUT.PUT_LINE('There is no Reservation with that
RESERVATION_ID');
    END IF;
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 trierów

Triggery:

- Trigger/triggery obsługujące:

- dodanie rezerwacji
- zmianę statusu

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`

Zadanie 5 - rozwiązanie

```
-- Trigger sprawdzający czy są wolne miejsca na wyjazd
create trigger TRG_CHECK_SEAT_AVAILABILITY_ADD_RESERVATION
    before insert
    on RESERVATION
    for each row
DECLARE
    v_available_seats NUMBER;
BEGIN
    Select MAX_NO_PLACES - (select count(*) from RESERVATION where
TRIP.TRIP_ID=RESERVATION.TRIP_ID and RESERVATION.STATUS!='C')
                into v_available_seats
            from TRIP
                where TRIP_ID=new.trip_id;

    IF v_available_seats <= 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'No available seats for this trip.');
    END IF;
END;
/

--Trigger sprawdzający czy wyjazd sie już nie odbył
CREATE OR REPLACE TRIGGER trg_check_date_add_reservation
BEFORE INSERT ON RESERVATION
FOR EACH ROW
DECLARE
    v_trip_date DATE;
BEGIN
    SELECT TRIP_DATE INTO v_trip_date
    FROM TRIP
    WHERE TRIP_ID = :NEW.trip_id;

    IF v_trip_date <= SYSDATE THEN
        DBMS_OUTPUT.PUT_LINE('Trip has already been completed.');
    END IF;
END;
```

```
-- Trigger sprawdzający czy są wolne miejsca lub czy nie odbył sie wyjazd dla
rezerwacji której zmieniamy status
create or replace trigger TRG_CHECK_SEAT_AVAILABILITY_MODIFY_STATUS
    before update of STATUS
    on RESERVATION
    for each row
DECLARE
    v_available_seats NUMBER;
    v_trip_date DATE;
BEGIN
    Select MAX_NO_PLACES - (select count(*) from RESERVATION where
TRIP.TRIP_ID=RESERVATION.TRIP_ID and STATUS!='C')
    into v_available_seats
    from TRIP
    where TRIP_ID=new.trip_id;
    IF v_available_seats <= 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Cannot modify status of a reservation for
a full trip.');
    END IF;
    SELECT TRIP_DATE INTO v_trip_date
    FROM TRIP
    WHERE TRIP_ID = OLD.TRIP_ID;

    IF v_trip_date <= SYSDATE THEN
        RAISE_APPLICATION_ERROR(-20002, 'Cannot modify status of a reservation for
ended trip.');
    END IF;
END;
/


/

-- Dodawanie Rezerwacji (nowsze)
create or replace PROCEDURE p_add_reservation_5(
    p_trip_id IN NUMBER,
    p_person_id IN NUMBER
)AS
BEGIN
    INSERT INTO RESERVATION (reservation_id, trip_id, person_id, status)
    VALUES ((SELECT MAX(reservation_id) + 1 FROM RESERVATION), p_trip_id,
p_person_id, 'P');
END;
/



-- Zmiana statusus Rezerwacji (nowsze)
CREATE OR REPLACE PROCEDURE p_modify_reservation_status_5(
    p_reservation_id IN NUMBER,
    p_status IN VARCHAR2
) AS
BEGIN
```

```
    UPDATE RESERVATION
    SET status = p_status
    WHERE reservation_id = p_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

```
alter table trip add
no_available_places int null;

create or replace procedure p_recalculate_available_places as
begin
```

```
    for record in (select trip_id, max_no_places from trip) loop
        update trip
        set no_available_places = record.max_no_places - (
            select count(*)
            from reservation
            where trip_id = record.trip_id
            and status = 'P'
        )
        where trip_id = record.trip_id;
    end loop;
    commit;
end;

begin
    p_recalculate_available_places;
end;


W tym zadaniu najpierw dodajemy pole no_available_places do tabeli trip,
następnie tworzymy procedure, która odejmuje od maksymalnej liczby miejsc w danej
wycieczce wszystkie miejsca, które są zajęte
przez rezerawcje potwierdzone i zapłacone. Wywołując tą procedurę jednokrotnie
nowo dodane pole w tabeli trip jest wypełniane wartościami
```

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

```
--dodawanie rezerwacji
```

```
create PROCEDURE p_add_reservation_6(
    p_trip_id IN trip.trip_id%TYPE,
    p_person_id IN reservation.person_id%TYPE
) AS
    v_max_places trip.max_no_places%TYPE;
BEGIN
    SELECT max_no_places INTO v_max_places FROM trip WHERE trip_id = p_trip_id;

    UPDATE trip
    SET no_available_places = no_available_places - 1
    WHERE trip_id = p_trip_id
    AND no_available_places > 0;

    INSERT INTO reservation (trip_id, person_id, status)
    VALUES (p_trip_id, p_person_id, 'N');
END;
/

--modyfikacja statusu rezerwacji
create PROCEDURE p_modify_reservation_status_6(
    p_reservation_id IN reservation.reservation_id%TYPE,
    p_new_status IN reservation.status%TYPE
) AS
    v_trip_id reservation.trip_id%TYPE;
BEGIN
    SELECT trip_id INTO v_trip_id FROM reservation WHERE reservation_id =
p_reservation_id;

    IF p_new_status = 'C' THEN
        UPDATE trip
        SET no_available_places = no_available_places + 1
        WHERE trip_id = v_trip_id;
    END IF;

    UPDATE reservation
    SET status = p_new_status
    WHERE reservation_id = p_reservation_id;

END;
/

--zmiana maksymalnej liczby miejsc wycieczki
CREATE PROCEDURE p_update_max_places(
    p_trip_id IN trip.trip_id%TYPE,
    p_new_max_places IN trip.max_no_places%TYPE
) AS
    v_diff_places NUMBER;
BEGIN
    SELECT p_new_max_places - max_no_places INTO v_diff_places
    FROM trip
    WHERE trip_id = p_trip_id;

    UPDATE trip
    SET no_available_places = no_available_places + v_diff_places,
```

```
        max_no_places = p_new_max_places
    WHERE trip_id = p_trip_id;

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 przy dodaniu nowej rezerwacji
create trigger TRG_RESERVATION_ADDED_6
    after insert
    on RESERVATION
    for each row
BEGIN
    UPDATE trip
    SET no_available_places = no_available_places - 1
    WHERE trip_id = :NEW.trip_id;
END;
/

--trigger przy odwołaniu rezerwacji
create trigger TRG_RESERVATION_STATUS_CHANGED_6
    after update of STATUS
    on RESERVATION
    for each row
BEGIN
    IF :OLD.status <> 'C' AND :NEW.status = 'C' THEN
        UPDATE trip
```

```
        SET no_available_places = no_available_places + 1
        WHERE trip_id = :NEW.trip_id;
    END IF;
END;
/
```

Zadanie 7 - podsumowanie

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

```
Oracle PL/SQL i MS Sqlserver T-SQL działają w ogólności podobnie i pozwalają na
programowanie relacyjnych baz danych, oraz interakcji wewnątrz nich.
Różnice porównamy na kilku płaszczyznach:
a) Typy danych:
Oracle PL/SQL posiada wsparcie dla typów prostych(np.
NUMBER, VARCHAR2, DATE), typów kolekcji (np. TABLE OF, VARRAY),
typów obiektowych(np. OBJECT), typów tablicy asocjacyjnej (np. PL/SQL TABLE)
T-SQL posiada wsparcie dla typów podobnych co Oracle (np. NUMBER, VARCHAR, DATE),
ale generalnie ich zasób jest nieco mniejszy, natomiast na bardziej złożone
struktury danych pozwala typ tabli(TABLE)
b) Funkcjonalność:
Oba języki wykorzystują podobne konstrukcje takie jak:
BEGIN ... END, IF ... ELSE, WHILE, CASE, LOOP
Oracle SQL ponadto posiada swoje unikalne konstrukcje takie jak: FORALL,
BULK COLLECT, EXCEPTION INIT
Natomiast T-SQL skupia się dodatkowo na danych zapisanych w SQL Server do których
daje nam dostęp funkcjami takimi jak : TOP, SET ROWCOUNT,
CTE (Common Table Expressions)
c) Funkcje analityczne:
W Oracle PL/SQL mamy funkcje takie jak: LAG, LEAD, ROW_NUMBER,
które umożliwiają wykonywanie zaawansowanych operacji
analitycznych bez konieczności pisania złożonych zapytań.
W T-SQL również mamy takie funkcję, chociaż ich zakres nieco się różni, są to np.
ROW_NUMBER(), RANK(), DENSE_RANK().
Wnioski:
Programowanie w PL/SQL oraz T-SQL co do zasady jest bardzo
podobne, ponieważ oba języki mają podobne zastosowanie,
różnice obawiają się w składni, dostępnych funkcjach oraz typach danych czy
praktykach takich jak zmienne globalne, których w PL/SQL stosuje się trochę więcej
Główna różnica jest natomiast platforma z której korzystamy
przy rozwiązaniach microsoftu prawdopodobnie łatwiej będzie
obsługiwać je bazą z T-SQL, natomiast w pozostałych OracleDatabase sprawdza się
równie dobrze
```