Oracle PL/Sql

widoki, funkcje, procedury, triggery ćwiczenie

Imiona i nazwiska autorów :

- Maciej Makowski
- Franciszek Job

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 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=214774 w szczególności dokument: 1_modyf.pdf

```
--INSERT
insert into trip(trip_name, country, trip_date, max_no_places)
values ('Słoneczna Ibiza', 'Hiszpania', to_date('2024-07-10','YYYY-MM-DD'), 4);
```

```
--UPDATE
update trip
set max_no_places=3
where trip_id = 21
```

```
--DELETE
delete trip
where trip_id = 4
```

```
--TRANSACTIONS
BEGIN
    SAVEPOINT sp;

    INSERT INTO person (PERSON_ID, firstname, lastname)
    VALUES (11, 'Jan', 'Owalski');
    INSERT INTO trip (trip_name, country, trip_date, max_no_places)
        VALUES ('Madera', 'Portugalia', TO_DATE('2026-05-01', 'YYYY-MM-DD'), 1);

    EXCEPTION
        WHEN OTHERS THEN
            ROLLBACK TO SAVEPOINT sp;
    COMMIT;
END;
```

- TRANSACTIONS

- Transakcja jest to sekwencja operacji wykonywana jako pojedyncza jednostka "pracy"
- commit- potwierdza transakcje(wykonuje operacjie z których się składa)
- rollback- wycofuje transakcje(nie wykonuje operacji z których się składa)
- W razie błędu zazwyczaj transakcje zostaje przerwana, tzn. zmiany przez nią wprowadzone nie zostają zatwierdzone, również możemy takie błędy obsługiwać specjalnym blokiem EXCEPTION

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

```
-- vw_reservation
create view VW_RESERVATION as
select r.RESERVATION_ID, t.COUNTRY, t.TRIP_DATE, t.TRIP_NAME,
       p.FIRSTNAME, p.LASTNAME, r.STATUS, r.TRIP_ID, r.PERSON_ID
from RESERVATION r
join TRIP t on t.TRIP_ID = r.TRIP_ID
join PERSON p on p.PERSON_ID = r.PERSON_ID;
```

```
-- vw_trip
create view VW_TRIP as
select trip.TRIP_ID, country, trip_date, trip_name,
max_no_places,
(max_no_places- count(reservation_id)) as no_available_places
from TRIP
left join RESERVATION
on TRIP.TRIP_ID = RESERVATION.TRIP_ID and RESERVATION.STATUS not like 'C'
group by TRIP.TRIP_ID, country, trip_date, trip_name, max_no_places
```

```
-- vw_available_trip
create view VW_AVAILABLE_TRIP as
select * from vw_trip
where TRIP_DATE > current_date 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

```
--f_trip_participants
create function f_trip_participants(tripID number )
    return trip_participants_table
as
    result trip_participants_table;
begin
    select TRIP_PARTICIPANT(p.PERSON_ID, p.FIRSTNAME, p.LASTNAME)
    bulk collect into result
    from RESERVATION r
    join PERSON p on p.PERSON_ID = r.PERSON_ID and r.STATUS not like 'C'
    where r.TRIP_ID = tripID;

    return result;
```

```
end;
```

```
--f_person_reservations
create function f_person_reservations(personID number )
    return person_reservations_table
as
    result person_reservations_table;
begin
    select person_reservation(r.RESERVATION_ID, r.TRIP_ID, r.PERSON_ID, r.STATUS)
    bulk collect into result
    from RESERVATION r
    where r.PERSON_ID = personID;
    return result;
end;
```

```
--f_available_trips_to
create FUNCTION f_available_trips_to(country_name VARCHAR2, date_from DATE, date_t
    RETURN trips_table
AS
    result trips_table;
BEGIN
      SELECT trip_data(t.TRIP_ID, t.TRIP_NAME, t.COUNTRY, t.TRIP_DATE, t.MAX_NO_PL
      BULK COLLECT INTO result
      FROM TRIP t
      WHERE t.COUNTRY LIKE country_name AND t.TRIP_DATE BETWEEN date_from AND date

      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_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

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

```
-- p_add_reservation
create procedure p_add_reservation(tripID in number, personID in number)
as
    v_available_places number;
begin
    select no_available_places into v_available_places
```

```
    from VW_AVAILABLE_TRIP
    where TRIP_ID = tripID
    GROUP BY no_available_places;

    insert into RESERVATION(reservation_id, trip_id, person_id, status)
    values (S_RESERVATION_SEQ.nextval, tripID, personID, 'N');

    insert into LOG(log_id, reservation_id, log_date, status)
    values(S_LOG_SEQ.nextval,S_RESERVATION_SEQ.currval, trunc(sysdate), 'N');

exception
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20004, 'Trip does not exist or ' ||
                                        'there are not any free places ' ||
                                         'or it has already taken place');
    when others then
        raise_application_error(-20003, 'Error inserting reservation: ' || SQLERRM
end p_add_reservation;
```

```
--p_modify_max_no_places
create PROCEDURE p_modify_max_no_places(tripID IN NUMBER, maxNoPlaces IN NUMBER)
AS
    v_reserved_places NUMBER;
BEGIN
    SELECT MAX_NO_PLACES - NO_AVAILABLE_PLACES INTO v_reserved_places
    FROM VW_TRIP
    WHERE TRIP_ID = tripID;

    IF maxNoPlaces < v_reserved_places THEN
        RAISE_APPLICATION_ERROR(-20002, 'It is not possible to change max_no_place
    END IF;

    UPDATE TRIP
    SET MAX_NO_PLACES = maxNoPlaces
    WHERE TRIP_ID = tripID;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20004, 'Trip does not exist');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20003, 'Error updating trip: ' || SQLERRM);
END p_modify_max_no_places;
```

```
-- p_modify_reservation_status
create PROCEDURE p_modify_reservation_status(
    p_reservation_id NUMBER,
    p_status CHAR
)
AS
    v_available_places NUMBER;
    v_trip_id NUMBER;
    v_current_status CHAR;
BEGIN
    SELECT TRIP_ID, STATUS INTO v_trip_id, v_current_status
    FROM RESERVATION WHERE RESERVATION_ID = p_reservation_id
    GROUP BY TRIP_ID, STATUS;

    SELECT NO_AVAILABLE_PLACES INTO v_available_places
```

```
    FROM vw_trip WHERE TRIP_ID = v_trip_id;

    IF v_available_places = 0 AND v_current_status = 'C' AND p_status != 'C' THEN
        RAISE_APPLICATION_ERROR(-20003, 'It is not possible to change status to a
    END IF;

    UPDATE RESERVATION SET STATUS = p_status
    WHERE RESERVATION_ID = p_reservation_id;

    INSERT INTO LOG (RESERVATION_ID, LOG_DATE, STATUS)
    VALUES (p_reservation_id, SYSDATE, p_status);

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20004, 'Trip or reservation does not exist.');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20005, 'An error occurred: ' || SQLERRM);
END p_modify_reservation_status;
```

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

```sql
--t_log_insert
create trigger T_LOG_INSERT
    after insert
    on RESERVATION
    for each row
begin
    insert into LOG(log_id, reservation_id, log_date, status)
    values(S_LOG_SEQ.nextval,:NEW.reservation_id, trunc(sysdate), 'N');
end;
```

```sql
--t_log_update
create trigger T_LOG_UPDATE
    after update
    on RESERVATION
    for each row
begin
    insert into LOG(log_id, reservation_id, log_date, status)
    values(S_LOG_SEQ.nextval,:NEW.reservation_id, trunc(sysdate), :NEW.status);
end;
```

```sql
--t_reservation_delete
create trigger t_reservation_delete
before delete on RESERVATION
for each row
begin
    RAISE_APPLICATION_ERROR(-20001, 'Cannot delete reservation.');
end;
```

```sql
--p_modify_reservation_status_4
create PROCEDURE p_modify_reservation_status_4(
    p_reservation_id NUMBER,
    p_status CHAR
)
AS
    v_available_places NUMBER;
    v_trip_id NUMBER;
    v_current_status CHAR;
BEGIN
    SELECT TRIP_ID, STATUS INTO v_trip_id, v_current_status
    FROM RESERVATION WHERE RESERVATION_ID = p_reservation_id
    GROUP BY TRIP_ID, STATUS;

    SELECT NO_AVAILABLE_PLACES INTO v_available_places
    FROM vw_trip WHERE TRIP_ID = v_trip_id;

    IF v_available_places = 0 AND v_current_status = 'C' AND p_status != 'C' THEN
        RAISE_APPLICATION_ERROR(-20003, 'It is not possible to change status to a
                                || 'than ''C'' when no available places are left.'
    END IF;

    UPDATE RESERVATION SET STATUS = p_status
    WHERE RESERVATION_ID = p_reservation_id;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
```

```
        RAISE_APPLICATION_ERROR(-20004, 'Trip or reservation does not exist.');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20005, 'An error occurred: ' || SQLERRM);
END p_modify_reservation_status_4;
```

```
--p_add_reservation_4
create procedure p_add_reservation_4(tripID in number, personID in number)
as
    v_available_places number;
begin
    select no_available_places into v_available_places
    from VW_AVAILABLE_TRIP
    where TRIP_ID = tripID
    GROUP BY no_available_places;

    insert into RESERVATION(reservation_id, trip_id, person_id, status)
    values (S_RESERVATION_SEQ.nextval, tripID, personID, 'N');

exception
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20004, 'Trip does not exist or ' ||
                                        'there are not any free places ' ||
                                         'or it has already taken place');
    when others then
        raise_application_error(-20003, 'Error inserting reservation: ' || SQLERRM
end p_add_reservation_4;
```

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

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

```
--t_before_insert_reservation
create trigger T_BEFORE_INSERT_RESERVATION
    before insert
    on RESERVATION
    for each row
DECLARE
    v_available_places NUMBER;
BEGIN
    SELECT no_available_places INTO v_available_places
    FROM VW_AVAILABLE_TRIP
    WHERE trip_id = :NEW.trip_id;
END;
```

```
--T_COMPOUND_BEFORE_UPDATE_RESERVATION
CREATE OR REPLACE TRIGGER T_COMPOUND_BEFORE_UPDATE_RESERVATION
FOR UPDATE ON RESERVATION
COMPOUND TRIGGER

    TYPE UPDATE_RESERVATION_STATUS_TABLE IS TABLE OF UPDATE_RESERVATION_STATUS;
    v_reservation_table UPDATE_RESERVATION_STATUS_TABLE := UPDATE_RESERVATION_STAT
    is_available NUMBER;

    BEFORE EACH ROW IS
    BEGIN
        v_reservation_table.EXTEND();
        v_reservation_table(v_reservation_table.LAST) :=
            UPDATE_RESERVATION_STATUS(:NEW.RESERVATION_ID, :NEW.TRIP_ID, :OLD.STAT
    END BEFORE EACH ROW;

    AFTER STATEMENT IS
    BEGIN
        FOR i IN 1..v_reservation_table.COUNT LOOP
            SELECT NO_AVAILABLE_PLACES INTO is_available FROM VW_TRIP
            WHERE TRIP_ID = v_reservation_table(i).TRIP_ID;

            IF is_available <= 0 AND v_reservation_table(i).OLD_STATUS = 'C' AND v
                RAISE_APPLICATION_ERROR(-20001,'It is not possible to change statu
            END IF;
        END LOOP;
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            RAISE_APPLICATION_ERROR(-20004, 'Trip or reservation does not exist.')
        WHEN OTHERS THEN
            RAISE_APPLICATION_ERROR(-20005, 'An error occurred: ' || SQLERRM);
    END AFTER STATEMENT;

END T_COMPOUND_BEFORE_UPDATE_RESERVATION;
```

```
-- p_add_reservation_5
create procedure p_add_reservation_5(tripID in number, personID in number)
as
begin
    insert into RESERVATION(reservation_id, trip_id, person_id, status)
    values (S_RESERVATION_SEQ.nextval, tripID, personID, 'N');
exception
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20004, 'Trip does not exist or ' ||
                                        'there are not any free places ' ||
                                         'or it has already taken place');
    when others then
        raise_application_error(-20003, 'Error inserting reservation: ' || SQLERRM
end p_add_reservation_5;
```

```
--p_modify_reservation_status_5
create PROCEDURE p_modify_reservation_status_5(
    p_reservation_id NUMBER,
    p_status CHAR
)
AS
    v_reservation_exist number;
BEGIN

    SELECT RESERVATION_ID INTO v_reservation_exist
    FROM RESERVATION WHERE RESERVATION_ID=p_reservation_id;

    UPDATE RESERVATION SET STATUS = p_status
    WHERE RESERVATION_ID = p_reservation_id;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20004, 'Trip or reservation does not exist.');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20005, 'An error occurred: ' || SQLERRM);
END p_modify_reservation_status_5;
```

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

```
--fill no_available_places column
declare
    v_available_places number := NULL;
begin
    for row_t in (SELECT * FROM TRIP) LOOP
        SELECT (max_no_places- count(reservation_id)) INTO v_available_places FROM
        left join RESERVATION
        on Trip.TRIP_ID = RESERVATION.TRIP_ID and RESERVATION.STATUS not like 'C'
        where TRIP.TRIP_ID = row_t.TRIP_ID
        group by max_no_places;

        UPDATE TRIP
        set no_available_places = v_available_places
        where trip_id = row_t.TRIP_ID;
    end loop;

    commit;
end;
```

```
--vw_available_trips_6
create view vw_available_trips_6
as
select * from TRIP
```

```
where TRIP_DATE>current_date and NO_AVAILABLE_PLACES>0
```

```
--f_available_trips_to_6
create FUNCTION f_available_trips_to_6(country_name VARCHAR2, date_from DATE, date
    RETURN trips_table_6
AS
    result trips_table_6;
BEGIN
      SELECT trip_data_6(t.TRIP_ID, t.TRIP_NAME, t.COUNTRY, t.TRIP_DATE, t.MAX_NO_
      BULK COLLECT INTO result
      FROM TRIP t
      WHERE t.COUNTRY LIKE country_name AND t.TRIP_DATE BETWEEN date_from AND date

      RETURN result;
END;

create type trip_data_6 as object(
    trip_id number,
    trip_name varchar2(100),
    country varchar2(50),
    trip_date date,
    max_no_places number,
    no_available_places number
)

create type TRIPS_TABLE_6 as table of TRIP_DATA_6
```

```
--p_modify_max_no_places_6
create or replace PROCEDURE p_modify_max_no_places_6(tripID IN NUMBER, maxNoPlaces
AS
    v_reserved_places NUMBER;
BEGIN
    SELECT MAX_NO_PLACES - NO_AVAILABLE_PLACES INTO v_reserved_places
    FROM TRIP
    WHERE TRIP_ID = tripID;

    IF maxNoPlaces < v_reserved_places THEN
        RAISE_APPLICATION_ERROR(-20002, 'It is not possible to change max_no_place
    END IF;

    UPDATE TRIP
    SET MAX_NO_PLACES = maxNoPlaces, NO_AVAILABLE_PLACES = maxNoPlaces - v_reserve
    WHERE TRIP_ID = tripID;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20004, 'Trip does not exist');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20003, 'Error updating trip: ' || SQLERRM);
END p_modify_max_no_places_6;
```

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

```
--p_add_reservation_6a
create procedure p_add_reservation_6a(tripID in number, personID in number)
as
    v_available_places number;
begin
    select no_available_places into v_available_places
    from VW_AVAILABLE_TRIPS_6
    where TRIP_ID = tripID;

    insert into RESERVATION(reservation_id, trip_id, person_id, status)
    values (S_RESERVATION_SEQ.nextval, tripID, personID, 'N');

    update trip
    set NO_AVAILABLE_PLACES = v_available_places-1
    where TRIP_ID = tripID;

exception
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20004, 'Trip does not exist or ' ||
                                        'there are not any free places ' ||
                                         'or it has already taken place');
    when others then
        raise_application_error(-20003, 'Error inserting reservation: ' || SQLERRM
end p_add_reservation_6a;
```

```
--p_modify_reservation_status_6a
create or replace PROCEDURE p_modify_reservation_status_6a(
    p_reservation_id NUMBER,
    p_status CHAR
)
AS
    v_available_places NUMBER;
    v_trip_id NUMBER;
    v_current_status CHAR;
BEGIN
    SELECT TRIP_ID, STATUS INTO v_trip_id, v_current_status
    FROM RESERVATION WHERE RESERVATION_ID = p_reservation_id
    GROUP BY TRIP_ID, STATUS;

    SELECT NO_AVAILABLE_PLACES INTO v_available_places
    FROM TRIP WHERE TRIP_ID = v_trip_id;

    IF v_available_places = 0 AND v_current_status = 'C' AND p_status != 'C' THEN
        RAISE_APPLICATION_ERROR(-20003, 'It is not possible to change status to a
                                || 'than ''C'' when no available places are left.'
    END IF;

    UPDATE RESERVATION SET STATUS = p_status
    WHERE RESERVATION_ID = p_reservation_id;

    IF v_current_status = 'C' AND p_status != 'C' THEN
        UPDATE TRIP
        SET NO_AVAILABLE_PLACES = v_available_places-1
        WHERE TRIP_ID = v_trip_id;
    ELSIF v_current_status != 'C' AND p_status = 'C' THEN
        UPDATE TRIP
        SET NO_AVAILABLE_PLACES = v_available_places+1
        WHERE TRIP_ID = v_trip_id;
    END IF;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20004, 'Trip or reservation does not exist.');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20005, 'An error occurred: ' || SQLERRM);
END p_modify_reservation_status_6a;
```

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

```
--t_before_insert_reservation_6b
create trigger t_before_insert_reservation_6b
    before insert
    on RESERVATION
    for each row
declare
    v_available_places number;
begin
    SELECT no_available_places INTO v_available_places
    FROM VW_AVAILABLE_TRIPS_6
    WHERE trip_id = :NEW.trip_id;

    UPDATE TRIP
    SET NO_AVAILABLE_PLACES = v_available_places-1
    WHERE TRIP_ID = :NEW.trip_id;

exception
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20004, 'Trip does not exist or ' ||
                                        'there are not any free places ' ||
                                         'or it has already taken place');
end;
```

```
--p_add_reservation_6b
create procedure p_add_reservation_6b(tripID in number, personID in number)
as
begin

    insert into RESERVATION(reservation_id, trip_id, person_id, status)
    values (S_RESERVATION_SEQ.nextval, tripID, personID, 'N');

exception
    when others then
        raise_application_error(-20003, 'Error inserting reservation: ' || SQLERRM
end p_add_reservation_6b;
```

```
--t_before_update_reservation_6b
create or replace trigger t_before_update_reservation_6b
    before update
    on RESERVATION
    for each row
```

```
declare
    v_available_places number;
begin
    SELECT NO_AVAILABLE_PLACES INTO v_available_places
    FROM TRIP WHERE TRIP_ID = :NEW.trip_id;

    IF v_available_places = 0 AND :OLD.status = 'C' AND :NEW.status != 'C' THEN
        RAISE_APPLICATION_ERROR(-20003, 'It is not possible to change status to a
                                || 'than ''C'' when no available places are left.'
    END IF;

    IF :OLD.status = 'C' AND :NEW.status != 'C' THEN
        UPDATE TRIP
        SET NO_AVAILABLE_PLACES = v_available_places-1
        WHERE TRIP_ID = :NEW.TRIP_ID;
    ELSIF :OLD.status != 'C' AND :NEW.status = 'C' THEN
        UPDATE TRIP
        SET NO_AVAILABLE_PLACES = v_available_places+1
        WHERE TRIP_ID = :NEW.trip_id;
    END IF;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20004, 'Trip does not exist.');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20005, 'An error occurred: ' || SQLERRM);
end;
```

```
--p_modify_reservation_status_6b
create PROCEDURE p_modify_reservation_status_6b(
    p_reservation_id NUMBER,
    p_status CHAR
)
AS
    v_reservation_exists number;
BEGIN
    SELECT count(*) INTO v_reservation_exists
    FROM RESERVATION WHERE RESERVATION_ID = p_reservation_id;

    IF v_reservation_exists = 0 THEN
        RAISE_APPLICATION_ERROR(-20003, 'Reservation does not exist');
    END IF;

    UPDATE RESERVATION SET STATUS = p_status
    WHERE RESERVATION_ID = p_reservation_id;

EXCEPTION
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20005, 'An error occurred: ' || SQLERRM);
END p_modify_reservation_status_6b;
```

Zadanie 7 - podsumowanie

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

1. Trudność

- PL/SQL - bardziej złozony
- T-SQL - stosunkowo prosty

2. Składnia:

- PL/SQL: bardziej deklaratywne podejście
- T-SQL: bardziej proceduralne podejście T-SQL uznawany za prostszy dla początkujących

3. Typy danych i zmienne

- typ money jest w T-SQL, brak w PL/SQL
- PL/SQL pozwala na definiowane typów danych uzytkownika, T-SQL tego nie zapewnia
- T-SQL ma opcję korzystania z tymczasowej tabeli, PL/SQL tego nie ma

4. Trigery

- PL/SQL: mogą być definiowane jako triger wierszowy (FOR EACH ROW) lub triger zbiorowy (FOR EACH STATEMENT).
- T-SQL: triger jest zawsze trigerem wierszowym i wykonuje się dla każdej zmiany wiersza, którą obejmuje.

5. Struktura bloków

- PL/SQL: zaczynają się od słowa kluczowego DECLARE i kończą na słowie kluczowym END.
- T-SQL: zaczynają się od BEGIN i kończą na END.

6. Dostęp do zmiennych specjalnych:

- PL/SQL: można korzystać ze zmiennych specjalnych, takich jak :NEW i :OLD, aby uzyskać dostęp do nowych i starych wartości wierszy.
- T-SQL: dostęp do nowych i starych wartości wierszy odbywa się za pomocą pseudo-tabeli INSERTED i DELETED.

7. Commitowanie

- PL/SQL: mozna uzyc AUTOCOMMIT

- T-SQL: brak komendy AUTOCOMMIT, trzeba manualnie commitować kazda transakcję

8. Gorsza osługa stringów w PL/SQL niz T-SQL
9. Lepsza obsługa zmiennych date/time w PL/SQL niz w T-SQL
10. Performance:
    - PL/SQL: szybszy czas wykonania złozonych zapytań, wydajniejszy przy większych ilościach danych
    - T-SQL: bardziej ograniczone uzycie pamięci systemowej (moze to sprawiac problemy przy pracy z większymi ilościami danych)
11. Obsługa błędów
    - PL/SQL: W PL/SQL można definiować obsługę błędów za pomocą bloków EXCEPTION.
    - T-SQL: W T-SQL obsługę błędów można zdefiniować za pomocą bloku TRY...CATCH.

Wnioski: Język PSL/SQL pozwala na więcej szczegółowości podczas pisania kodu w stosunku do T-SQL. Technologia pochodząca od Oracle pozwala takze na wykonanie wielu deklaracji SQL na raz po zamknięciu ich w bloku. PL/SQL posiada równiez dobre wsparcie obsługi błędów. Jego składnia moze być jednak na początku trochę trudniejsza w przyswajaniu jego zawiłości.