Oracle PL/Sql

widoki, funkcje, procedury, triggery ćwiczenie

Imiona i nazwiska autorów : Jakub Zając, Szymon Borusiewicz

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 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=311899 w szczególności dokument: `1_ora_modyf.pdf`

```
ALTER TABLE reservation ADD no_tickets INT DEFAULT 1 NOT NULL;

ALTER TABLE log ADD no_tickets INT DEFAULT 1 NOT NULL;

BEGIN
    INSERT INTO reservation (trip_id, person_id, status, no_tickets)
    VALUES (3, 3, 'N', 2);

    INSERT INTO log (reservation_id, log_date, status, no_tickets)
    VALUES (s_reservation_seq.currval, SYSDATE, 'N', 2);

    COMMIT;
END;

begin
    insert into person (firstname, lastname)
    values ('Maciej', 'Antoniuk');
    insert into person (firstname, lastname)
    values ('Wiktor', 'Szczepaniak');

    rollback;
end;

select * from person;

begin
    insert into person (firstname, lastname)
    values ('Maciej', 'Antoniuk');
    insert into person (firstname, lastname)
    values ('Wiktor', 'Szczepaniak');

    commit;
end;
```

Po wykonaniu commit, wszystkie zmiany są zapisane w bazie i stają się widoczne dla wszystkich użytkowników.

Po wykonaniu rollback baza wraca do stanu sprzed rozpoczęcia transakcji. Jeśli transakcja w Oracle jest wykonywana i napotka błąd, wykona się automatycznie rollback.

W Oracle, w przeciwieństwie do MS SQL Server, autocommity są domyślnie wyłączone. W Oracle można używać obsługi wyjątków EXCEPTION, podczas gdy w MS SQL Server błędy obsługuje się przy użyciu TRY i CATCH.

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

```
CREATE OR REPLACE VIEW vw_reservation AS
SELECT
    RESERVATION_ID,
    COUNTRY,
    TRIP_DATE,
    TRIP_NAME,
    FIRSTNAME,
    LASTNAME,
    STATUS,
    RESERVATION.TRIP_ID,
```

```
    RESERVATION.PERSON_ID,
    NO_TICKETS
FROM RESERVATION
JOIN TRIP ON RESERVATION.TRIP_ID = TRIP.TRIP_ID
JOIN PERSON ON RESERVATION.PERSON_ID = PERSON.PERSON_ID;

CREATE OR REPLACE VIEW vw_trip AS
SELECT
    T.TRIP_ID,
    T.COUNTRY,
    T.TRIP_DATE,
    T.TRIP_NAME,
    T.MAX_NO_PLACES,
    (T.MAX_NO_PLACES - COALESCE(SUM(R.NO_TICKETS), 0)) AS no_available_places
FROM TRIP T
LEFT JOIN RESERVATION R ON T.TRIP_ID = R.TRIP_ID AND STATUS IN ('N', 'P')
GROUP BY T.TRIP_ID, T.COUNTRY, T.TRIP_DATE, T.TRIP_NAME, T.MAX_NO_PLACES;

CREATE OR REPLACE VIEW vw_available_trip AS
SELECT
    T.TRIP_ID,
    T.COUNTRY,
    T.TRIP_DATE,
    T.TRIP_NAME,
    T.MAX_NO_PLACES,
    (T.MAX_NO_PLACES - COALESCE(SUM(R.NO_TICKETS), 0)) AS no_available_places
FROM TRIP T
LEFT JOIN RESERVATION R ON T.TRIP_ID = R.TRIP_ID AND STATUS IN ('N', 'P')
WHERE T.TRIP_DATE > CURRENT_DATE
GROUP BY T.TRIP_ID, T.COUNTRY, T.TRIP_DATE, T.TRIP_NAME, T.MAX_NO_PLACES
HAVING (T.MAX_NO_PLACES - COALESCE(SUM(R.NO_TICKETS), 0)) > 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

```
CREATE OR REPLACE TYPE t_reservation_view_row AS OBJECT (
    reservation_id INT,
    country VARCHAR2(50),
    trip_date DATE,
    trip_name VARCHAR2(100),
    firstname VARCHAR2(50),
    lastname VARCHAR2(50),
    status CHAR(1),
    trip_id INT,
    person_id INT,
    no_tickets INT
);

CREATE OR REPLACE TYPE t_reservation_view_table AS TABLE OF t_reservation_view_row;

CREATE OR REPLACE FUNCTION f_trip_participants(trip_id IN INT)
RETURN t_reservation_view_table PIPELINED AS
BEGIN
    FOR rec IN (
        SELECT
            R.RESERVATION_ID,
            T.COUNTRY,
            T.TRIP_DATE,
            T.TRIP_NAME,
            P.FIRSTNAME,
            P.LASTNAME,
            R.STATUS,
            R.TRIP_ID,
            R.PERSON_ID,
            R.NO_TICKETS
        FROM RESERVATION R
        JOIN TRIP T ON R.TRIP_ID=T.TRIP_ID
        JOIN PERSON P ON R.PERSON_ID=P.PERSON_ID
```

```
        WHERE R.TRIP_ID = f_trip_participants.trip_id
    )
    LOOP
        PIPE ROW (t_reservation_view_row(
                rec.RESERVATION_ID, rec.COUNTRY, rec.TRIP_DATE,
                rec.TRIP_NAME, rec.FIRSTNAME, rec.LASTNAME,
                rec.STATUS, rec.TRIP_ID, rec.PERSON_ID,
                rec.NO_TICKETS
                  ));
    end loop;
    return;
end;

SELECT * FROM TABLE(f_trip_participants(1));

CREATE OR REPLACE FUNCTION f_person_reservations(person_id IN INT)
RETURN t_reservation_view_table PIPELINED AS
BEGIN
    FOR rec IN (
        SELECT
            R.RESERVATION_ID,
            T.COUNTRY,
            T.TRIP_DATE,
            T.TRIP_NAME,
            P.FIRSTNAME,
            P.LASTNAME,
            R.STATUS,
            R.TRIP_ID,
            R.PERSON_ID,
            R.NO_TICKETS
        FROM RESERVATION R
        JOIN TRIP T ON R.TRIP_ID=T.TRIP_ID
        JOIN PERSON P ON R.PERSON_ID=P.PERSON_ID
        WHERE R.PERSON_ID = f_person_reservations.PERSON_ID
    )
    LOOP
        PIPE ROW (t_reservation_view_row(
                rec.RESERVATION_ID, rec.COUNTRY, rec.TRIP_DATE,
                rec.TRIP_NAME, rec.FIRSTNAME, rec.LASTNAME,
                rec.STATUS, rec.TRIP_ID, rec.PERSON_ID,
                rec.NO_TICKETS
                  ));
    end loop;
    return;
end;

SELECT * FROM TABLE(f_person_reservations(10));

CREATE OR REPLACE TYPE trip_view_row AS OBJECT (
    trip_id INT,
    trip_name VARCHAR2(100),
    country VARCHAR2(50),
    trip_date DATE,
    max_no_places INT
);

CREATE OR REPLACE TYPE trip_view_table AS TABLE OF trip_view_row;

CREATE OR REPLACE FUNCTION f_available_trips_to(
    country IN VARCHAR2,
    date_from IN DATE,
    date_to IN DATE
)
RETURN trip_view_table PIPELINED AS
BEGIN
    FOR rec IN (
        SELECT T.TRIP_ID, T.TRIP_NAME, T.COUNTRY, T.TRIP_DATE, T.MAX_NO_PLACES
        FROM TRIP T
        WHERE T.country = f_available_trips_to.country
        AND T.TRIP_DATE BETWEEN date_from AND date_to
    )
    LOOP
        PIPE ROW (trip_view_row(
            rec.TRIP_ID, rec.TRIP_NAME, rec.COUNTRY, rec.TRIP_DATE, rec.MAX_NO_PLACES
        ));
    end loop;
    return;
end;

SELECT * FROM TABLE(f_available_trips_to('Polska', TO_DATE('2025-01-01', 'YYYY-MM-DD'), TO_DATE('2025-12-31', 'YYYY-MM-DD')));
```

Kontrola parametrów pozwala zapobiegać błędom (np. wywołaniu funkcji z nieistniejącym parametrem) czy może poprawić wydajność przez ograniczenie zbędnych zapytań. Wadą jest konieczność pisania dodatkowego kodu do obsługi wyjątków.

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

```
CREATE OR REPLACE PROCEDURE p_add_reservation(
    p_trip_id IN RESERVATION.trip_id%TYPE,
    p_person_id IN RESERVATION.person_id%TYPE,
    p_no_ticket IN NUMBER
) AS
    v_max_places TRIP.max_no_places%TYPE;
    v_reserved_places NUMBER;
    v_trip_date TRIP.trip_date%TYPE;
BEGIN
    SELECT MAX_NO_PLACES, TRIP_DATE INTO v_max_places, v_trip_date
    FROM TRIP WHERE TRIP_ID = p_trip_id;

    IF v_trip_date < SYSDATE THEN
        RAISE_APPLICATION_ERROR(-20001, 'Nie można zarezerwować miejsca na wycieczkę, która już się odbyła');
    end if;

    SELECT COALESCE(SUM(no_tickets), 0) INTO v_reserved_places FROM RESERVATION WHERE TRIP_ID = p_trip_id AND status != 'C';;

    IF v_reserved_places + p_no_ticket > v_max_places THEN
        RAISE_APPLICATION_ERROR(-20002, 'Brak wystarczającej liczby miejsc na wycieczkę');
    end if;

    INSERT INTO RESERVATION(trip_id, person_id, status)
    VALUES (p_trip_id, p_person_id, 'N');

    INSERT INTO log (reservation_id, LOG_DATE, status)
    VALUES (S_RESERVATION_SEQ.currval, SYSDATE, 'N');

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20003, 'Nie znalziono wycieczki');
    WHEN OTHERS THEN
        ROLLBACK;
        RAISE;
end;

CREATE OR REPLACE PROCEDURE p_modify_reservation_status(
    p_reservation_id IN RESERVATION.reservation_id%TYPE,
    p_status IN RESERVATION.status%TYPE
) AS
    v_trip_id RESERVATION.trip_id%TYPE;
    v_max_places TRIP.max_no_places%TYPE;
    v_reserved_places NUMBER;
    v_old_status RESERVATION.status%TYPE;
BEGIN
    SELECT trip_id, status INTO v_trip_id, v_old_status
    FROM reservation WHERE reservation_id = p_reservation_id;

    SELECT max_no_places INTO v_max_places FROM TRIP WHERE trip_id = v_trip_id;
    SELECT COALESCE(SUM(no_tickets), 0) INTO v_reserved_places FROM RESERVATION WHERE trip_id = v_trip_id AND status != 'C';

    IF v_old_status = 'C' AND v_reserved_places >= v_max_places THEN
        RAISE_APPLICATION_ERROR(-20004, 'Brak miejsc na wycieczkę');
    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(-20005, 'Nie znaleziono rezerwacji');
    WHEN OTHERS THEN
        ROLLBACK;
        RAISE;
end;

CREATE OR REPLACE PROCEDURE p_modify_reservation(
    p_reservation_id IN RESERVATION.reservation_id%TYPE,
    p_no_tickets IN NUMBER
) AS
    v_trip_id RESERVATION.trip_id%TYPE;
    v_max_places TRIP.max_no_places%TYPE;
    v_reserved_places NUMBER;
    v_current_tickets NUMBER;
BEGIN
    SELECT trip_id, no_tickets INTO v_trip_id, v_current_tickets FROM RESERVATION WHERE reservation_id = p_reservation_id;
    SELECT max_no_places INTO v_max_places FROM TRIP WHERE trip_id = v_trip_id;
    SELECT COALESCE(SUM(no_tickets), 0) INTO v_reserved_places FROM RESERVATION WHERE trip_id = v_trip_id AND status != 'C';
```

```
    IF v_reserved_places - v_current_tickets + p_no_tickets > v_max_places THEN
        RAISE_APPLICATION_ERROR(-20006, 'Brak wystarczającej liczby miejsc na wycieczkę');
    end if;

    UPDATE RESERVATION SET no_tickets = p_no_tickets WHERE reservation_id = p_reservation_id;

    INSERT INTO log (reservation_id, log_date, status)
    VALUES (p_reservation_id, SYSDATE, 'P');

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20007, 'Nie znaleziono rezerwacji');
    WHEN OTHERS THEN
        ROLLBACK;
        RAISE;
end;

CREATE OR REPLACE PROCEDURE p_modify_max_no_places(
    p_trip_id IN TRIP.trip_id%TYPE,
    p_max_no_places IN TRIP.max_no_places%TYPE
) AS
    v_reserved_places NUMBER;
BEGIN
    SELECT COALESCE(SUM(no_tickets), 0) INTO v_reserved_places FROM RESERVATION WHERE trip_id = p_trip_id AND status != 'C';

    IF p_max_no_places < v_reserved_places THEN
        RAISE_APPLICATION_ERROR(-20008, 'Nie można zmniejszyć liczby miejsc poniżej liczby zarezerwowanych miejsc');
    end if;

    UPDATE TRIP SET max_no_places = p_max_no_places WHERE trip_id = p_trip_id;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20009, 'Nie znaleziono wycieczki');
    WHEN OTHERS THEN
        ROLLBACK;
        RAISE;
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

```
CREATE OR REPLACE TRIGGER trg_log_add_reservation
AFTER INSERT ON RESERVATION
FOR EACH ROW
BEGIN
    INSERT INTO log (reservation_id, log_date, status)
    VALUES (:NEW.reservation_id, SYSDATE, :NEW.status);
end;

CREATE OR REPLACE TRIGGER trg_log_modify_status
AFTER UPDATE OF status ON RESERVATION
FOR EACH ROW
WHEN (OLD.status != NEW.status)
BEGIN
    INSERT INTO log (reservation_id, log_date, status)
    VALUES (:NEW.reservation_id, SYSDATE, :NEW.status);
end;

CREATE OR REPLACE TRIGGER trg_log_modify_tickets
AFTER UPDATE OF no_tickets ON RESERVATION
FOR EACH ROW
WHEN (OLD.no_tickets != NEW.no_tickets)
BEGIN
    INSERT INTO log (reservation_id, log_date, status)
    VALUES (:NEW.reservation_id, SYSDATE, 'Tickets updated');
end;

CREATE OR REPLACE TRIGGER trg_prevent_delete_reservation
BEFORE DELETE ON RESERVATION
FOR EACH ROW
BEGIN
    RAISE_APPLICATION_ERROR(-20010, 'Usunięcie rezerwacji jest zabronione');
end;

CREATE OR REPLACE PROCEDURE p_add_reservation_4(
    p_trip_id IN RESERVATION.trip_id%TYPE,
    p_person_id IN RESERVATION.person_id%TYPE,
```

```
    p_no_ticket IN NUMBER
) AS
    v_max_places TRIP.max_no_places%TYPE;
    v_reserved_places NUMBER;
    v_trip_date TRIP.trip_date%TYPE;
BEGIN
    SELECT MAX_NO_PLACES, TRIP_DATE INTO v_max_places, v_trip_date
    FROM TRIP WHERE TRIP_ID = p_trip_id;

    IF v_trip_date < SYSDATE THEN
        RAISE_APPLICATION_ERROR(-20001, 'Nie można zarezerwować miejsca na wycieczkę, która już się odbyła');
    end if;

    SELECT COALESCE(SUM(no_tickets), 0) INTO v_reserved_places FROM RESERVATION WHERE TRIP_ID = p_trip_id AND status != 'C';;

    IF v_reserved_places + p_no_ticket > v_max_places THEN
        RAISE_APPLICATION_ERROR(-20002, 'Brak wystarczającej liczby miejsc na wycieczkę');
    end if;

    INSERT INTO RESERVATION(trip_id, person_id, status)
    VALUES (p_trip_id, p_person_id, 'N');

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20003, 'Nie znalziono wycieczki');
    WHEN OTHERS THEN
        ROLLBACK;
        RAISE;
end;

CREATE OR REPLACE PROCEDURE p_modify_reservation_status_4(
    p_reservation_id IN RESERVATION.reservation_id%TYPE,
    p_status IN RESERVATION.status%TYPE
) AS
    v_trip_id RESERVATION.trip_id%TYPE;
    v_max_places TRIP.max_no_places%TYPE;
    v_reserved_places NUMBER;
    v_old_status RESERVATION.status%TYPE;
BEGIN
    SELECT trip_id, status INTO v_trip_id, v_old_status
    FROM RESERVATION WHERE reservation_id = p_reservation_id;

    SELECT max_no_places INTO v_max_places FROM TRIP WHERE trip_id = v_trip_id;
    SELECT COALESCE(SUM(no_tickets), 0) INTO v_reserved_places FROM RESERVATION WHERE trip_id = v_trip_id AND status != 'C';

    IF v_old_status = 'C' AND v_reserved_places >= v_max_places THEN
        RAISE_APPLICATION_ERROR(-20004, 'Brak miejsc na wycieczkę');
    end if;

    UPDATE RESERVATION SET status = p_status WHERE reservation_id = p_reservation_id;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20005, 'Nie znaleziono rezerwacji');
    WHEN OTHERS THEN
        ROLLBACK;
        RAISE;
end;

CREATE OR REPLACE PROCEDURE p_modify_reservation_4(
    p_reservation_id IN RESERVATION.reservation_id%TYPE,
    p_no_tickets IN NUMBER
) AS
    v_trip_id RESERVATION.trip_id%TYPE;
    v_max_places TRIP.max_no_places%TYPE;
    v_reserved_places NUMBER;
    v_current_tickets NUMBER;
BEGIN
    SELECT trip_id, no_tickets INTO v_trip_id, v_current_tickets FROM RESERVATION WHERE reservation_id = p_reservation_id;
    SELECT max_no_places INTO v_max_places FROM TRIP WHERE trip_id = v_trip_id;
    SELECT COALESCE(SUM(no_tickets), 0) INTO v_reserved_places FROM RESERVATION WHERE trip_id = v_trip_id AND status != 'C';

    IF v_reserved_places - v_current_tickets + p_no_tickets > v_max_places THEN
        RAISE_APPLICATION_ERROR(-20006, 'Brak wystarczającej liczby miejsc na wycieczkę');
    end if;

    UPDATE RESERVATION SET no_tickets = p_no_tickets WHERE reservation_id = p_reservation_id;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20007, 'Nie znaleziono rezerwacji');
    WHEN OTHERS THEN
        ROLLBACK;
        RAISE;
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

```
CREATE OR REPLACE TRIGGER trg_check_available_places
BEFORE INSERT ON RESERVATION
FOR EACH ROW
DECLARE
    v_max_places TRIP.max_no_places%TYPE;
    v_reserved_places NUMBER;
BEGIN
    SELECT max_no_places INTO v_max_places FROM TRIP WHERE trip_id = :NEW.trip_id;

    SELECT COALESCE(SUM(no_tickets), 0) INTO v_reserved_places
    FROM RESERVATION
    WHERE trip_id = :NEW.trip_id AND status != 'C';

    IF v_reserved_places + :NEW.no_tickets > v_max_places THEN
        RAISE_APPLICATION_ERROR(-20010, 'Brak wystarczającej liczby miejsc na wycieczkę');
    end if;
end;

CREATE OR REPLACE TRIGGER trg_check_status_update
BEFORE UPDATE of status ON RESERVATION
FOR EACH ROW
WHEN (NEW.status IN ('N', 'P'))
DECLARE
    v_max_places TRIP.max_no_places%TYPE;
    v_reserved_places NUMBER;
BEGIN
    SELECT max_no_places INTO v_max_places FROM TRIP WHERE trip_id = :NEW.trip_id;

    SELECT COALESCE(SUM(no_tickets), 0) INTO v_reserved_places
    FROM RESERVATION
    WHERE trip_id = :NEW.trip_id AND status != 'C';

    IF v_reserved_places > v_max_places THEN
        RAISE_APPLICATION_ERROR(-20011, 'Brak miejsc na wycieczkę');
    end if;
end;

CREATE OR REPLACE TRIGGER trg_check_ticket_update
BEFORE UPDATE of no_tickets ON RESERVATION
FOR EACH ROW
DECLARE
    v_max_places TRIP.max_no_places%TYPE;
    v_reserved_places NUMBER;
BEGIN
    SELECT max_no_places INTO v_max_places FROM TRIP WHERE trip_id = :NEW.trip_id;

    SELECT COALESCE(SUM(no_tickets), 0) INTO v_reserved_places
    FROM RESERVATION
    WHERE trip_id = :NEW.trip_id AND status != 'C';

    IF v_reserved_places - :OLD.no_tickets + :NEW.no_tickets > v_max_places THEN
        RAISE_APPLICATION_ERROR(-20012, 'Brak miejsc na wycieczkę');
    end if;
end;

CREATE OR REPLACE PROCEDURE p_add_reservation_5(
    p_trip_id IN RESERVATION.trip_id%TYPE,
    p_person_id IN RESERVATION.person_id%TYPE,
    p_no_ticket IN NUMBER
) AS
    v_trip_date TRIP.trip_date%TYPE;
BEGIN
    SELECT TRIP_DATE INTO v_trip_date
    FROM TRIP WHERE TRIP_ID = p_trip_id;

    IF v_trip_date < SYSDATE THEN
        RAISE_APPLICATION_ERROR(-20001, 'Nie można zarezerwować miejsca na wycieczkę, która już się odbyła');
    end if;

    INSERT INTO RESERVATION(trip_id, person_id, NO_TICKETS, status)
    VALUES (p_trip_id, p_person_id, p_no_ticket,'N');

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20003, 'Nie znalziono wycieczki');
    WHEN OTHERS THEN
        ROLLBACK;
        RAISE;
end;

CREATE OR REPLACE PROCEDURE p_modify_reservation_status_5(
    p_reservation_id IN RESERVATION.reservation_id%TYPE,
    p_status IN RESERVATION.status%TYPE
) AS
    v_trip_id RESERVATION.trip_id%TYPE;
    v_old_status RESERVATION.status%TYPE;
BEGIN
    SELECT trip_id, status INTO v_trip_id, v_old_status
    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(-20005, 'Nie znaleziono rezerwacji');
    WHEN OTHERS THEN
        ROLLBACK;
```

```
        RAISE;
end;

CREATE OR REPLACE PROCEDURE p_modify_reservation_5(
    p_reservation_id IN RESERVATION.reservation_id%TYPE,
    p_no_tickets IN NUMBER
) AS
BEGIN
    UPDATE RESERVATION SET no_tickets = p_no_tickets WHERE reservation_id = p_reservation_id;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20007, 'Nie znaleziono rezerwacji');
    WHEN OTHERS THEN
        ROLLBACK;
        RAISE;
end;

BEGIN
    p_add_reservation_5(3, 9, 1);
END;

select * from VW_RESERVATION;

BEGIN
    p_modify_reservation_status_5(42, 'P');
end;

BEGIN
    p_modify_reservation_5(42, 2);
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_no_available_places_6 AS
BEGIN
    UPDATE TRIP T
    SET T.no_available_places = T.max_no_places -
        COALESCE((SELECT SUM(R.no_tickets)
                  FROM RESERVATION R
                  WHERE R.trip_id = T.trip_id AND R.status IN ('N', 'P')), 0);
END;

BEGIN
    p_recalculate_no_available_places_6;
end;

CREATE OR REPLACE TRIGGER tr_add_reservation_6_before_insert
BEFORE INSERT ON RESERVATION
FOR EACH ROW
DECLARE
    v_available_places NUMBER;
    v_trip_date DATE;
BEGIN
    SELECT trip_date, no_available_places INTO v_trip_date, v_available_places
    FROM TRIP WHERE trip_id = :NEW.trip_id FOR UPDATE;

    IF v_trip_date < SYSDATE THEN
        RAISE_APPLICATION_ERROR(-20002, 'Nie można zarezerwować miejsc na wycieczkę, która już się odbyła');
    end if;

    IF v_available_places < :NEW.no_tickets THEN
        RAISE_APPLICATION_ERROR(-20001, 'Brak wystarczających miejsc na wycieczkę');
    end if;

    UPDATE TRIP
    SET no_available_places = no_available_places - :NEW.no_tickets
    WHERE trip_id = :NEW.trip_id;
end;

CREATE OR REPLACE TRIGGER tr_modify_reservation_6_after_update
AFTER UPDATE OF no_tickets ON RESERVATION
FOR EACH ROW
```

```
BEGIN
    UPDATE TRIP
    SET TRIP.no_available_places = TRIP.no_available_places - (:NEW.no_tickets - :OLD.no_tickets)
    WHERE trip_id = :NEW.trip_id;
end;

CREATE OR REPLACE TRIGGER tr_modify_reservation_status_6_before_update
BEFORE UPDATE OF STATUS ON RESERVATION
FOR EACH ROW
DECLARE
    v_available_places NUMBER;
BEGIN
    IF :OLD.status IN ('N', 'P') AND :NEW.status = 'C' THEN
        UPDATE TRIP
        SET no_available_places = no_available_places + :OLD.no_tickets
        WHERE trip_id = :OLD.trip_id;

    ELSIF :OLD.status = 'C' AND :NEW.status IN ('N', 'P') THEN
        SELECT no_available_places INTO v_available_places
        FROM TRIP WHERE trip_id = :NEW.trip_id;

        IF v_available_places < :NEW.no_tickets THEN
            RAISE_APPLICATION_ERROR(-20006, 'Brak wystarczających miejsc na przywrócenie rezerwacji');
        end if;

        UPDATE TRIP
        SET TRIP.no_available_places = TRIP.no_available_places - :NEW.no_tickets
        WHERE trip_id = :NEW.trip_id;
    end if;
end;

CREATE OR REPLACE PROCEDURE p_add_reservation_6(
    p_trip_id IN RESERVATION.trip_id%TYPE,
    p_person_id IN RESERVATION.person_id%TYPE,
    p_no_tickets IN NUMBER
) AS
BEGIN
    INSERT INTO RESERVATION(trip_id, person_id, no_tickets, status)
    VALUES(p_trip_id, p_person_id, p_no_tickets, 'N');
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20003, 'Nie znaleziono wycieczki');
end;

CREATE OR REPLACE PROCEDURE p_modify_reservation_status_6(
    p_reservation_id IN RESERVATION.reservation_id%TYPE,
    p_status IN RESERVATION.status%TYPE
) AS
BEGIN
    UPDATE RESERVATION SET status = p_status WHERE reservation_id = p_reservation_id;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20004, 'Nie znaleziono rezerwacji');
end;

CREATE OR REPLACE PROCEDURE p_modify_reservation_6(
    p_reservation_id IN RESERVATION.reservation_id%TYPE,
    p_no_tickets IN NUMBER
) AS
BEGIN
    UPDATE RESERVATION SET no_tickets = p_no_tickets WHERE reservation_id = p_reservation_id;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20005, 'Nie znaleziono rezerwacji');
end;

SELECT * FROM VW_RESERVATION;
SELECT * FROM VW_AVAILABLE_TRIP;

BEGIN
    p_add_reservation_6(3, 6, 1);
END;

BEGIN
    p_modify_reservation_status_6(63, 'C');
end;
```

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

```
ALTER TRIGGER tr_add_reservation_6_before_insert DISABLE;
ALTER TRIGGER tr_modify_reservation_6_after_update DISABLE;
ALTER TRIGGER tr_modify_reservation_status_6_before_update DISABLE;

CREATE OR REPLACE PROCEDURE p_add_reservation_6a(
    p_trip_id IN RESERVATION.trip_id%TYPE,
    p_person_id IN RESERVATION.person_id%TYPE,
```

```
    p_no_tickets IN NUMBER
) AS
    v_available_places NUMBER;
    v_trip_date DATE;
BEGIN
    SELECT trip_date, no_available_places INTO v_trip_date, v_available_places
    FROM TRIP WHERE trip_id = p_trip_id FOR UPDATE;

    IF v_trip_date < SYSDATE THEN
        RAISE_APPLICATION_ERROR(-20002, 'Nie można zarezerwować miejsc na wycieczkę, która już się odbyła');
    end if;

    IF v_available_places < p_no_tickets THEN
        RAISE_APPLICATION_ERROR(-20001, 'Brak wystarczających miejsc na wycieczkę');
    end if;

    INSERT INTO RESERVATION(trip_id, person_id, no_tickets, status)
    VALUES(p_trip_id, p_person_id, p_no_tickets, 'N');

    UPDATE TRIP
    SET no_available_places = no_available_places - p_no_tickets
    WHERE trip_id = p_trip_id;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20003, 'Nie znaleziono wycieczki');
end;

CREATE OR REPLACE PROCEDURE p_modify_reservation_status_6a(
    p_reservation_id IN RESERVATION.reservation_id%TYPE,
    p_status IN RESERVATION.status%TYPE
) AS
    v_trip_id RESERVATION.trip_id%TYPE;
    v_no_tickets RESERVATION.no_tickets%TYPE;
    v_old_status RESERVATION.status%TYPE;
    v_available_places NUMBER;
BEGIN
    SELECT trip_id, no_tickets, status INTO v_trip_id, v_no_tickets, v_old_status
    FROM RESERVATION WHERE reservation_id = p_reservation_id FOR UPDATE;

    UPDATE RESERVATION SET status = p_status WHERE reservation_id = p_reservation_id;

    IF v_old_status IN ('N', 'P') AND p_status = 'C' THEN
        UPDATE TRIP SET no_available_places = no_available_places + v_no_tickets
        WHERE trip_id = v_trip_id;
    ELSIF v_old_status = 'C' AND p_status IN ('N', 'P') THEN
        SELECT no_available_places INTO v_available_places FROM TRIP
        WHERE trip_id = v_trip_id;

        IF v_available_places < v_no_tickets THEN
            RAISE_APPLICATION_ERROR(-20006, 'Brak wystarczających miejsc na przywrócenie rezerwacji');
        end if;

        UPDATE TRIP SET no_available_places = no_available_places - v_no_tickets
        WHERE trip_id = v_trip_id;
    end if;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20004, 'Nie znaleziono rezerwacji');
end;

CREATE OR REPLACE PROCEDURE p_modify_reservation_6a(
    p_reservation_id IN RESERVATION.reservation_id%TYPE,
    p_no_tickets IN NUMBER
) AS
    v_trip_id RESERVATION.trip_id%TYPE;
    v_old_no_tickets RESERVATION.no_tickets%TYPE;
    v_status RESERVATION.status%TYPE;
    v_available_places NUMBER;
BEGIN
    SELECT trip_id, no_tickets, status INTO v_trip_id, v_old_no_tickets, v_status
    FROM RESERVATION WHERE reservation_id = p_reservation_id FOR UPDATE;

    SELECT no_available_places INTO v_available_places FROM TRIP WHERE trip_id = v_trip_id;

    IF p_no_tickets > v_old_no_tickets AND v_available_places < (p_no_tickets - v_old_no_tickets) THEN
        RAISE_APPLICATION_ERROR(-20009, 'Brak wystarczającej liczby miejsc na wycieczkę');
    end if;

    UPDATE RESERVATION SET no_tickets = p_no_tickets WHERE reservation_id = p_reservation_id;

    UPDATE TRIP SET no_available_places = no_available_places - (p_no_tickets - v_old_no_tickets)
    WHERE trip_id = v_trip_id;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20005, 'Nie znaleziono rezerwacji');
end;

CREATE OR REPLACE PROCEDURE p_modify_max_no_places_6a(
    p_trip_id IN TRIP.trip_id%TYPE,
    p_new_max_no_places IN TRIP.max_no_places%TYPE
) AS
    v_reserved_places NUMBER;
BEGIN
    SELECT SUM(no_tickets) INTO v_reserved_places
    FROM RESERVATION WHERE trip_id = p_trip_id AND status IN ('N', 'P');

    IF p_new_max_no_places < v_reserved_places THEN
        RAISE_APPLICATION_ERROR(-20007, 'Nowa maksymalna liczba miejc nie może być mniejsza niż liczba już zarezerwowanych miejsc');
    end if;

    UPDATE TRIP SET max_no_places = p_new_max_no_places WHERE trip_id = p_trip_id;

    p_recalculate_no_available_places_6;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20008, 'Nie znaleziono wycieczki');
end;
```

```
SELECT * FROM VW_AVAILABLE_TRIP;
SELECT * FROM VW_RESERVATION;

BEGIN
    p_modify_max_no_places_6a(3, 8);
END;
BEGIN
    p_add_reservation_6a(3, 13, 2);
END;

BEGIN
    p_modify_reservation_status_6a(81, 'P');
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

```
ALTER TRIGGER tr_add_reservation_6_before_insert DISABLE;
ALTER TRIGGER tr_modify_reservation_6_after_update DISABLE;
ALTER TRIGGER tr_modify_reservation_status_6_before_update DISABLE;

CREATE OR REPLACE TRIGGER tr_add_reservation_6b_before_insert
BEFORE INSERT ON RESERVATION
FOR EACH ROW
DECLARE
    v_available_places NUMBER;
    v_trip_date DATE;
BEGIN
    SELECT trip_date, no_available_places INTO v_trip_date, v_available_places
    FROM TRIP WHERE trip_id = :NEW.trip_id FOR UPDATE;

    IF v_trip_date < SYSDATE THEN
        RAISE_APPLICATION_ERROR(-20002, 'Nie można zarezerwować miejsc na wycieczkę, która już się odbyła');
    end if;

    IF v_available_places < :NEW.no_tickets THEN
        RAISE_APPLICATION_ERROR(-20001, 'Brak wystarczających miejsc na wycieczkę');
    end if;

    UPDATE TRIP
    SET no_available_places = no_available_places - :NEW.no_tickets
    WHERE trip_id = :NEW.trip_id;
end;

CREATE OR REPLACE TRIGGER tr_modify_reservation_6b_after_update
AFTER UPDATE OF no_tickets ON RESERVATION
FOR EACH ROW
BEGIN
    UPDATE TRIP
    SET no_available_places = no_available_places - (:NEW.no_tickets - :OLD.no_tickets)
    WHERE trip_id = :NEW.trip_id;
end;

CREATE OR REPLACE TRIGGER tr_modify_reservation_status_6b_before_update
BEFORE UPDATE OF status ON RESERVATION
FOR EACH ROW
DECLARE
    v_available_places NUMBER;
BEGIN
    IF :OLD.status IN ('N', 'P') AND :NEW.status = 'C' THEN
        UPDATE TRIP
        SET no_available_places = no_available_places + :OLD.no_tickets
        WHERE trip_id = :OLD.trip_id;
    ELSIF :OLD.status = 'C' AND :NEW.status IN ('N', 'P') THEN
        SELECT no_available_places INTO v_available_places FROM TRIP
        WHERE trip_id = :NEW.trip_id;

        IF v_available_places < :NEW.no_tickets THEN
            RAISE_APPLICATION_ERROR(-20006, 'Brak wystarczającej liczby miejsc na przywrócenie rezerwacji');
        end if;

        UPDATE TRIP
        SET no_available_places = no_available_places - :NEW.no_tickets
        WHERE trip_id = :NEW:trip_id;
    end if;
end;

CREATE OR REPLACE PROCEDURE p_add_reservation_6b(
    p_trip_id IN RESERVATION.trip_id%TYPE,
    p_person_id IN RESERVATION.person_id%TYPE,
    p_no_tickets IN NUMBER
) AS
BEGIN
    INSERT INTO RESERVATION(trip_id, person_id, no_tickets, status)
    VALUES(p_trip_id, p_person_id, p_no_tickets, 'N');
EXCEPTION
    WHEN NO_DATA_FOUND THEN
```

```
        RAISE_APPLICATION_ERROR(-20003, 'Nie znaleziono wycieczki');
end;

CREATE OR REPLACE PROCEDURE p_modify_reservation_status_6b(
    p_reservation_id IN RESERVATION.reservation_id%TYPE,
    p_status IN RESERVATION.status%TYPE
) AS
BEGIN
    UPDATE RESERVATION SET status = p_status WHERE reservation_id = p_reservation_id;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20004, 'Nie znaleziono rezerwacji');
end;

CREATE OR REPLACE PROCEDURE p_modify_reservation_6b(
    p_reservation_id IN RESERVATION.reservation_id%TYPE,
    p_no_tickets IN NUMBER
) AS
BEGIN
    UPDATE RESERVATION SET no_tickets = p_no_tickets WHERE reservation_id = p_reservation_id;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20005, 'Nie znaleziono rezerwacji');
end;
```

Zadanie 7 - podsumowanie

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

W Oracle inaczej odbywa się obsługa wyjątków (przy użyciu EXCEPTION), podczas gdy w MS SQL Server używa się TRY i CATCH. W Oracle zmienne definiuje się bez "@" w przeciwieństwie do MS SQL Server. Oracle lepiej zarządza współbieżnością niż MS SQL Server, co w przypadku MS SQL Server może prowadzić do większej liczby konfliktów przy równoczesnym dostępie do danych.