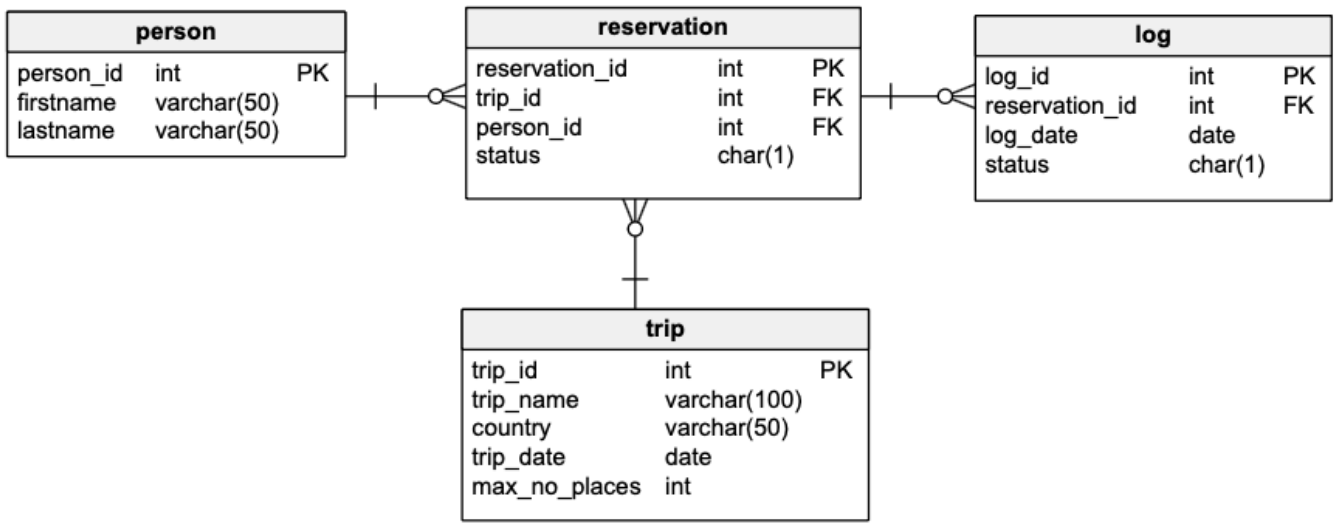
Oracle PL/Sql

widoki, funkcje, procedury, triggery ćwiczenie

Imiona i nazwiska autorów : Kluza Łukasz Mateusz Sacha

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 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 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`

```
-- person
insert into person(firstname, lastname)
values ('Ala', 'Nowakowska');

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

insert into person(firstname, lastname)
values ('Anna', 'Kowalczyk');

insert into person(firstname, lastname)
values  ('Katarzyna', 'Wojciechowska');

insert into person(firstname, lastname)
values ('Andrzej ', 'Kaczmarek');
```

```sql
insert into person(firstname, lastname)
values  ('Marek ', 'Kwiatkowski');

-- reservation
-- trip 1
insert  into reservation(trip_id, person_id, status)
values (1, 5, 'N');

-- trip 2
insert into reservation(trip_id, person_id, status)
values (2, 9, 'C');

-- trip 3
insert into reservation(trip_id, person_id, status)
values (4, 10, 'N');

-- trip 4
insert into reservation(trip_id, person_id, status)
values (4, 7, 'C');

insert into reservation(trip_id, person_id, status)
values (4, 2, 'C');

insert into reservation(trip_id, person_id, status)
values (4, 8, 'P');
```

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

```
create or replace view vw_reservation as
select reservation_id, country, trip_date, trip_name, firstname, lastname, 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



create or replace view vw_trip as
with reservation_counter as (select trip_id, count(*) counter
                             from reservation r_in
                             where status <> 'C'
                             group by trip_id)

select t.trip_id,
       country,
       trip_date,
       trip_name,
       max_no_places,
       max_no_Places-counter no_available_places
from trip t
        left join reservation_counter rc on rc.trip_id = t.trip_id



create or replace view vw_available_trip as
select * from vw_trip
where no_available_places > 0 and trip_date > sysdate
```

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

- 2.1 *f_trip_participants*

```
create or replace type trip_participants as OBJECT
(
    trip_id        int,
    reservation_id int,
    country        varchar(50),
    trip_date      date,
    trip_name      varchar(100),
    firstname      varchar(50),
    lastname       varchar(50),
    status         char(1),
    person_id      int
);

create or replace type trip_participants_table is table of trip_participants;

create or replace function f_trip_participants(trip_id int)
    return trip_participants_table
as
    result trip_participants_table;
    counter int;
begin
    select count(*)
    into counter
    from trip t
```

```
    where t.trip_id = f_trip_participants.trip_id;

    if counter = 0 then
        raise_application_error(-20001, 'Trip ' || f_trip_participants.trip_id
        || ' does not exist');
    end if;

    select trip_participants(vwr.trip_id, vwr.reservation_id, vwr.country,
        vwr.trip_date, vwr.trip_name, vwr.firstname, vwr.lastname,
        vwr.status, vwr.person_id) bulk collect
    into result
    from vw_reservation vwr
    where vwr.trip_id = f_trip_participants.trip_id
      and vwr.status = 'P';

    return result;
end;
```

- 2.2 *f_person_reservations*

```
create or replace type person_reservation as OBJECT
(
    trip_id        int,
    reservation_id int,
    country        varchar(50),
    trip_date      date,
    trip_name      varchar(100),
    firstname      varchar(50),
    lastname       varchar(50),
    status         char(1),
    person_id      int
);

create or replace type person_reservations_table is table of person_reservation;

create or replace function f_person_reservations(person_id int)
    return person_reservations_table
as
    result person_reservations_table;
    counter int;
begin
    select count(*)
    into counter
    from person p
    where p.person_id = f_person_reservations.person_id;

    if counter = 0 then
        raise_application_error(-20001, 'Person '
        || f_person_reservations.person_id || ' does not exist');
    end if;

    select person_reservation(vwr.trip_id, vwr.reservation_id, vwr.country,
```

```
        vwr.trip_date, vwr.trip_name, vwr.firstname, vwr.lastname,
        vwr.status, vwr.person_id) bulk collect
    into result
    from vw_reservation vwr
    where vwr.person_id = f_person_reservations.person_id;
    return result;
end;
```

2.3 f_available_trips_to

```
create or replace type country_trip as OBJECT
(
    trip_id        int,
    country        varchar(50),
    trip_date      date,
    trip_name      varchar(100)
);

CREATE OR REPLACE FUNCTION f_available_trips_to(
    country_name VARCHAR2,
    date_from DATE,
    date_to DATE
)
RETURN country_trip_table
AS
    result country_trip_table := country_trip_table();
    counter INT := 0;
BEGIN
    SELECT COUNT(*)
    INTO counter
    FROM trip t
    WHERE t.country = country_name
        AND t.trip_date >= date_from AND t.trip_date <= date_to;

    IF counter = 0 THEN
        raise_application_error(-20001, 'Trip to ' || country_name
        || ' does not exist between: ' || date_from ||' and ' || date_to);
    END IF;

    SELECT country_trip(t.trip_id, t.country, t.trip_date, t.trip_name)
    BULK COLLECT INTO result
    FROM trip t
    WHERE t.country = country_name
        AND t.trip_date >= date_from AND t.trip_date <= 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

- 3.1 *p_add_reservation*

```
CREATE OR REPLACE PROCEDURE p_add_reservation(
    p_trip_id INT,
    p_person_id INT
)
AS
    v_trip_date DATE;
    v_no_places INT;
    v_reservation_id INT;
    v_trip_exists INT;
    v_person_exists INT;
BEGIN

    SELECT COUNT(*) INTO v_trip_exists
    FROM trip
    WHERE trip_id = p_trip_id;

    IF v_trip_exists = 0 THEN
        RAISE_APPLICATION_ERROR(-20003, 'Trip does not exist.');
    END IF;

    SELECT COUNT(*) INTO v_person_exists
    FROM person
    WHERE person_id = p_person_id;

    IF v_person_exists = 0 THEN
        RAISE_APPLICATION_ERROR(-20004, 'Person does not exist.');
    END IF;

    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, 'This trip has already taken place.');
    END IF;

    SELECT max_no_places - NVL((SELECT COUNT(*) FROM reservation
                                WHERE trip_id = p_trip_id), 0)
    INTO v_no_places
    FROM trip
    WHERE trip_id = p_trip_id;

    IF v_no_places <= 0 THEN
        RAISE_APPLICATION_ERROR(-20002, 'There are no available seats for this
trip.');
    END IF;

    INSERT INTO reservation(trip_id, person_id, status)
    VALUES (p_trip_id, p_person_id, 'N')
    RETURNING reservation_id INTO v_reservation_id;

    INSERT INTO log(reservation_id,log_date,status)
    VALUES (v_reservation_id, SYSDATE, 'N');
```

- 3.2 *p_modify_reservation_status*

```
CREATE OR REPLACE PROCEDURE p_modify_reservation_status(
    p_reservation_id INT,
    p_status CHAR
)
AS
    v_status CHAR;
    v_no_places INT;
    v_trip_id INT;
    v_reservation_exists INT;
BEGIN

    SELECT COUNT(*) INTO v_reservation_exists
    FROM reservation
    WHERE reservation_id = p_reservation_id;

    IF v_reservation_exists = 0 THEN
        RAISE_APPLICATION_ERROR(-20005, 'Reservation does not exist.');
    END IF;

    SELECT status,trip_id INTO v_status,v_trip_id
    FROM reservation
    WHERE reservation_id = p_reservation_id;

    SELECT max_no_places - NVL((SELECT COUNT(*) FROM reservation
                                WHERE trip_id = v_trip_id), 0)
    INTO v_no_places
    FROM trip
    WHERE trip_id = v_trip_id;

    IF v_status = 'C' or (p_status = 'N' and v_status = 'P') THEN
        RAISE_APPLICATION_ERROR(-20001, 'It is no possible
        to change the status reservation');
    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);
END;
```

- 3.3 *p_modify_max_no_places*

```
CREATE OR REPLACE PROCEDURE p_modify_max_no_places(
    p_trip_id INT,
    p_max_no_places INT
```

```
)
AS
    v_current_reservations INT;
BEGIN
    SELECT COUNT(*)
    INTO v_current_reservations
    FROM trip
    WHERE trip_id = p_trip_id;

    IF v_current_reservations = 0 THEN
        RAISE_APPLICATION_ERROR(-20002, 'Trip with specified trip_id
                                does not exist.');
    END IF;

    SELECT COUNT(*)
    INTO v_current_reservations
    FROM reservation
    WHERE trip_id = p_trip_id;

    IF p_max_no_places < v_current_reservations THEN
        RAISE_APPLICATION_ERROR(-20001, 'New maximum number
        of places cannot be less than current reservations.');
    END IF;

    UPDATE trip
    SET max_no_places = p_max_no_places
    WHERE trip_id = p_trip_id;

END;
```

Zadanie 4 - triggery

Zmiana strategii zapisywania do dziennika rezerwacji. Realizacja przy pomocy triggerów

Należy wprowadzić zmianę, która spowoduje, że zapis do dziennika 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

- 4.1 *trg_reservation_insert*

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

- 4.2 *trg_reservation_status_update*

```
CREATE OR REPLACE TRIGGER trg_reservation_status_update
AFTER UPDATE OF status ON reservation
FOR EACH ROW
BEGIN
    INSERT INTO log(reservation_id, log_date, status)
    VALUES (:OLD.reservation_id, SYSDATE, :NEW.status);
END;
```

- 4.3 *trg_reservation_delete*

```
CREATE OR REPLACE TRIGGER trg_reservation_delete
BEFORE DELETE ON reservation
FOR EACH ROW
BEGIN
    RAISE_APPLICATION_ERROR(-20001, 'Deleting reservations is not allowed.');
END;
```

- 4.4 *p_add_reservation_4*

```
CREATE OR REPLACE PROCEDURE p_add_reservation_4(
    p_trip_id INT,
    p_person_id INT
)
AS
```

```
    v_trip_date DATE;
    v_no_places INT;
    v_reservation_id INT;
    v_trip_exists INT;
    v_person_exists INT;
BEGIN

    SELECT COUNT(*) INTO v_trip_exists
    FROM trip
    WHERE trip_id = p_trip_id;

    IF v_trip_exists = 0 THEN
        RAISE_APPLICATION_ERROR(-20003, 'Trip does not exist.');
    END IF;

    SELECT COUNT(*) INTO v_person_exists
    FROM person
    WHERE person_id = p_person_id;

    IF v_person_exists = 0 THEN
        RAISE_APPLICATION_ERROR(-20004, 'Person does not exist.');
    END IF;

    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, 'This trip has already taken place.');
    END IF;

    SELECT max_no_places - NVL((SELECT COUNT(*) FROM reservation
                                WHERE trip_id = p_trip_id), 0)
    INTO v_no_places
    FROM trip
    WHERE trip_id = p_trip_id;

    IF v_no_places <= 0 THEN
        RAISE_APPLICATION_ERROR(-20002, 'There are no available seats for this
trip.');
    END IF;

    INSERT INTO reservation(trip_id, person_id, status)
    VALUES (p_trip_id, p_person_id, 'N')
    RETURNING reservation_id INTO v_reservation_id;

END;
```

- 4.5 *p_modify_reservation_status_4*

```
CREATE OR REPLACE PROCEDURE p_modify_reservation_status_4(
    p_reservation_id INT,
```

```
    p_status CHAR
)
AS
    v_status CHAR;
    v_no_places INT;
    v_trip_id INT;
    v_reservation_exists INT;
BEGIN

    SELECT COUNT(*) INTO v_reservation_exists
    FROM reservation
    WHERE reservation_id = p_reservation_id;

    IF v_reservation_exists = 0 THEN
        RAISE_APPLICATION_ERROR(-20005, 'Reservation does not exist.');
    END IF;

    SELECT status,trip_id INTO v_status,v_trip_id
    FROM reservation
    WHERE reservation_id = p_reservation_id;

    SELECT max_no_places - NVL((SELECT COUNT(*) FROM reservation
                                WHERE trip_id = v_trip_id), 0)
    INTO v_no_places
    FROM trip
    WHERE trip_id = v_trip_id;

    IF v_status = 'C' or (p_status = 'N' and v_status = 'P') THEN
        RAISE_APPLICATION_ERROR(-20001, 'It is no possible to change the status
reservation');
    END IF;

    UPDATE reservation
    SET status = p_status
    WHERE reservation_id = p_reservation_id;

END;
```

Zadanie 5 - triggery

Zmiana strategii kontroli dostępności miejsc. Realizacja przy pomocy triggerów

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

- 5.1 *trg_add_reservation*

```
CREATE OR REPLACE TRIGGER trg_add_reservation
BEFORE INSERT ON reservation
FOR EACH ROW
DECLARE
    v_max_no_places INT;
    v_reserved_places INT;
BEGIN

    SELECT max_no_places INTO v_max_no_places
    FROM trip
    WHERE trip_id = :NEW.trip_id;

    SELECT COUNT(*) INTO v_reserved_places
    FROM reservation
    WHERE trip_id = :NEW.trip_id;

    IF v_reserved_places >= v_max_no_places THEN
        RAISE_APPLICATION_ERROR(-20001, 'There are no available seats for this
trip.');
    END IF;
END;
```

- 5.2 *trg_modify_reservation_status*

```
CREATE OR REPLACE TRIGGER trg_modify_reservation_status
BEFORE UPDATE OF status ON reservation
FOR EACH ROW
DECLARE
    v_max_no_places INT;
    v_reserved_places INT;
BEGIN
```

```
    SELECT max_no_places INTO v_max_no_places
    FROM trip
    WHERE trip_id = :NEW.trip_id;

    SELECT COUNT(*) INTO v_reserved_places
    FROM reservation
    WHERE trip_id = :NEW.trip_id;

    IF v_reserved_places >= v_max_no_places and :OLD.status = 'C'
        and :NEW.status = 'P' THEN
        RAISE_APPLICATION_ERROR(-20001, 'There are no available seats for this
trip.');
    END IF;
END;
```

- 5.3 *p_add_reservation_5*

```
CREATE OR REPLACE PROCEDURE p_add_reservation_5(
    p_trip_id INT,
    p_person_id INT
)
AS
    v_trip_date DATE;
    v_no_places INT;
    v_reservation_id INT;
    v_trip_exists INT;
    v_person_exists INT;
BEGIN

    SELECT COUNT(*) INTO v_trip_exists
    FROM trip
    WHERE trip_id = p_trip_id;

    IF v_trip_exists = 0 THEN
        RAISE_APPLICATION_ERROR(-20003, 'Trip does not exist.');
    END IF;

    SELECT COUNT(*) INTO v_person_exists
    FROM person
    WHERE person_id = p_person_id;

    IF v_person_exists = 0 THEN
        RAISE_APPLICATION_ERROR(-20004, 'Person does not exist.');
    END IF;

    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, 'This trip has already taken place.');
```

```
    END IF;

    INSERT INTO reservation(trip_id, person_id, status)
    VALUES (p_trip_id, p_person_id, 'N')
    RETURNING reservation_id INTO v_reservation_id;

END;
```

- 5.4 *p_modify_reservation_status_5*

```
CREATE OR REPLACE PROCEDURE p_modify_reservation_status_5(
    p_reservation_id INT,
    p_status CHAR
)
AS
    v_status CHAR;
    v_no_places INT;
    v_trip_id INT;
    v_reservation_exists INT;
BEGIN

    SELECT COUNT(*) INTO v_reservation_exists
    FROM reservation
    WHERE reservation_id = p_reservation_id;

    IF v_reservation_exists = 0 THEN
        RAISE_APPLICATION_ERROR(-20005, 'Reservation does not exist.');
    END IF;

    SELECT status,trip_id INTO v_status,v_trip_id
    FROM reservation
    WHERE reservation_id = p_reservation_id;

    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

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

- 6a 1.no_available_places

```
create or replace procedure no_available_places is
begin
```

```
    UPDATE trip t1
SET no_available_places = (select count(person_id)
                           from vw_reservation vwr
                           where vwr.status <> 'C' and t1.trip_id = vwr.trip_id);
end;
```

- 6a 2.p_add_reservation_6

```
create or replace PROCEDURE p_add_reservation_6a(
    p_trip_id INT,
    p_person_id INT
)
AS
    v_trip_date DATE;
    v_no_places INT;
    v_reservation_id INT;
    v_trip_exists INT;
    v_person_exists INT;
BEGIN

    SELECT COUNT(*) INTO v_trip_exists
    FROM trip
    WHERE trip_id = p_trip_id;

    IF v_trip_exists = 0 THEN
        RAISE_APPLICATION_ERROR(-20003, 'Trip does not exist.');
    END IF;

    SELECT COUNT(*) INTO v_person_exists
    FROM person
    WHERE person_id = p_person_id;

    IF v_person_exists = 0 THEN
        RAISE_APPLICATION_ERROR(-20004, 'Person does not exist.');
    END IF;

    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, 'This trip has already taken place.');
    END IF;

    SELECT MAX_NO_PLACES - NO_AVAILABLE_PLACES
    INTO v_no_places
    FROM trip
    WHERE trip_id = p_trip_id;

    IF v_no_places <= 0 THEN
        RAISE_APPLICATION_ERROR(-20002, 'There are no available seats for this
trip.');
```

```
    END IF;

    INSERT INTO reservation(trip_id, person_id, status)
    VALUES (p_trip_id, p_person_id, 'N')
    RETURNING reservation_id INTO v_reservation_id;

    no_available_places();
    DBMS_OUTPUT.PUT_LINE('Reservation added successfully.');
END;
```

- 6a 3.p_modify_max_no_places_6a

```
create or replace PROCEDURE p_modify_max_no_places_6a(
    p_trip_id INT,
    p_max_no_places INT
)
AS
    v_current_reservations INT;
BEGIN
    SELECT COUNT(*)
    INTO v_current_reservations
    FROM trip
    WHERE trip_id = p_trip_id;

    IF v_current_reservations = 0 THEN
        RAISE_APPLICATION_ERROR(-20002, 'Trip with specified trip_id does not
exist.');
    END IF;

    SELECT t.no_available_places
    INTO v_current_reservations
    FROM trip t
    WHERE trip_id = p_trip_id;

    IF p_max_no_places < v_current_reservations THEN
        RAISE_APPLICATION_ERROR(-20001, 'New maximum number of places cannot be
        less than current reservations.');
    END IF;

    UPDATE trip
    SET max_no_places = p_max_no_places
    WHERE trip_id = p_trip_id;

    DBMS_OUTPUT.PUT_LINE('Maximum number of places updated successfully.');
    NO_AVAILABLE_PLACES();
END;
```

- 6a 4.p_modify_reservation_status_6a

```
create or replace PROCEDURE p_modify_reservation_status_6a(
    p_reservation_id INT,
    p_status CHAR
)
AS
    v_status CHAR;
    v_trip_id INT;
    v_reservation_exists INT;
BEGIN

    SELECT COUNT(*) INTO v_reservation_exists
    FROM reservation
    WHERE reservation_id = p_reservation_id;

    IF v_reservation_exists = 0 THEN
        RAISE_APPLICATION_ERROR(-20005, 'Reservation does not exist.');
    END IF;

    SELECT status,trip_id INTO v_status,v_trip_id
    FROM reservation
    WHERE reservation_id = p_reservation_id;

    IF v_status = 'C' or (p_status = 'N' and v_status = 'P') THEN
        RAISE_APPLICATION_ERROR(-20001, 'It is no possible to change the status
reservation');
    END IF;

    UPDATE reservation
    SET status = p_status
    WHERE reservation_id = p_reservation_id;
    NO_AVAILABLE_PLACES();
    DBMS_OUTPUT.PUT_LINE('Reservation status updated successfully.');
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

- 6b 1. trg_reservation_insert_6B

```
create or replace trigger TRG_RESERVATION_INSERT_6B
    after insert
    on RESERVATION
    for each row
BEGIN
    INSERT INTO log(reservation_id, log_date, status)
    VALUES (:NEW.reservation_id, SYSDATE, :NEW.status);
    NO_AVAILABLE_PLACES();
END;
```

- 6b 2. trg_reservation_status_update_6B

```
create or replace trigger TRG_RESERVATION_STATUS_UPDATE_6B
    after update of STATUS
    on RESERVATION
    for each row
BEGIN
    INSERT INTO log(reservation_id, log_date, status)
    VALUES (:OLD.reservation_id, SYSDATE, :NEW.status);
    NO_AVAILABLE_PLACES();
END;
```

- 6b 3. p_add_reservation_6b

```
create or replace PROCEDURE p_add_reservation_6b(
    p_trip_id INT,
    p_person_id INT
)
AS
    v_trip_date DATE;
    v_no_places INT;
    v_reservation_id INT;
    v_trip_exists INT;
    v_person_exists INT;
BEGIN

    SELECT COUNT(*) INTO v_trip_exists
    FROM trip
    WHERE trip_id = p_trip_id;

    IF v_trip_exists = 0 THEN
```

```
        RAISE_APPLICATION_ERROR(-20003, 'Trip does not exist.');
    END IF;

    SELECT COUNT(*) INTO v_person_exists
    FROM person
    WHERE person_id = p_person_id;

    IF v_person_exists = 0 THEN
        RAISE_APPLICATION_ERROR(-20004, 'Person does not exist.');
    END IF;

    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, 'This trip has already taken place.');
    END IF;

    SELECT MAX_NO_PLACES - NO_AVAILABLE_PLACES
    INTO v_no_places
    FROM trip
    WHERE trip_id = p_trip_id;

    IF v_no_places <= 0 THEN
        RAISE_APPLICATION_ERROR(-20002, 'There are no available seats for this
trip.');
    END IF;

    INSERT INTO reservation(trip_id, person_id, status)
    VALUES (p_trip_id, p_person_id, 'N')
    RETURNING reservation_id INTO v_reservation_id;

    DBMS_OUTPUT.PUT_LINE('Reservation added successfully.');
END;
```

- 6b 4. p_modify_max_no_places_6b

```
create or replace PROCEDURE p_modify_max_no_places_6b(
    p_trip_id INT,
    p_max_no_places INT
)
AS
    v_current_reservations INT;
BEGIN
    SELECT COUNT(*)
    INTO v_current_reservations
    FROM trip
    WHERE trip_id = p_trip_id;

    IF v_current_reservations = 0 THEN
        RAISE_APPLICATION_ERROR(-20002, 'Trip with specified trip_id does not
```

```
exist.');
    END IF;

    SELECT t.no_available_places
    INTO v_current_reservations
    FROM trip t
    WHERE trip_id = p_trip_id;

    IF p_max_no_places < v_current_reservations THEN
        RAISE_APPLICATION_ERROR(-20001, 'New maximum number of places cannot be
        less than current reservations.');
    END IF;

    UPDATE trip
    SET max_no_places = p_max_no_places
    WHERE trip_id = p_trip_id;

    DBMS_OUTPUT.PUT_LINE('Maximum number of places updated successfully.');
END;
```

- 6b 4. p_modify_reservation_status_6b

```
create or replace PROCEDURE p_modify_reservation_status_6b(
    p_reservation_id INT,
    p_status CHAR
)
AS
    v_status CHAR;
    v_trip_id INT;
    v_reservation_exists INT;
BEGIN

    SELECT COUNT(*) INTO v_reservation_exists
    FROM reservation
    WHERE reservation_id = p_reservation_id;

    IF v_reservation_exists = 0 THEN
        RAISE_APPLICATION_ERROR(-20005, 'Reservation does not exist.');
    END IF;

    SELECT status,trip_id INTO v_status,v_trip_id
    FROM reservation
    WHERE reservation_id = p_reservation_id;

    IF v_status = 'C' or (p_status = 'N' and v_status = 'P') THEN
        RAISE_APPLICATION_ERROR(-20001, 'It is no possible to change the status
reservation');
    END IF;

    UPDATE reservation
    SET status = p_status
    WHERE reservation_id = p_reservation_id;
```

```
    DBMS_OUTPUT.PUT_LINE('Reservation status updated successfully.');
END;
```

Zadanie 7 - podsumowanie

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

```
Programowanie w Oracle PL/SQL i Microsoft SQL Server T-SQL różni się głównie
składnią i funkcjonalnościami:

1. PL/SQL: Opiera się na języku programowania proceduralnego, z blokami kodu
BEGIN...END.
T-SQL: Ma bardziej zwięzłą składnię, zbliżoną do standardowego języka zapytań SQL.

2. PL/SQL: Deklaracje zmiennych na początku bloku, różnorodne typy danych.
T-SQL: Mniejszy wybór typów danych, proste deklaracje zmiennych.

3. PL/SQL: Rozbudowany mechanizm obsługi wyjątków.
T-SQL: Obsługuje wyjątki za pomocą instrukcji TRY...CATCH.

4. PL/SQL: Obsługuje transakcje za pomocą COMMIT i ROLLBACK.
T-SQL: Również wspiera transakcje, ale z inną składnią.

5. PL/SQL: Posiada funkcje do manipulacji danymi daty i czasu, takie jak SYSDATE,
TO_DATE.
T-SQL: Również oferuje funkcje do pracy z danymi daty i czasu, jednak o innych
nazwach
takie jak GETDATE(), CONVERT.

6. PL/SQL: Przy tworzeniu procedur, konieczne jest zdefiniowanie obiektów
typu tabela przed ich użyciem w procedurze.
T-SQL: Nie ma potrzeby definiowania obiektów typu tabela przed użyciem ich w
procedurze.
```