

Oracle PL/Sql

widoki, funkcje, procedury, triggerzy ćwiczenie

Imiona i nazwiska autorów : Krzysztof Swędzioł, Piotr Błaszczuk

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

pomocne mogą być materiały dostępne tu:

<https://upel.agh.edu.pl/mod/folder/view.php?id=214774> w szczególności dokument: `1_modyf.pdf`

```
-- Polecenie commit powoduje zatwierdzenie przeprowadzanych zmian w bloku
BEGIN...END, COMMIT; dzięki niemu zmiany są tworzone na stałe a nie tylko
--na czas działania obecnej sesji. rollback natomiast cofa ostatnio utworzone
zmiany aż do ostatniego SavePointa ustawionego przez programistę. Pierwszy screen
przedstawia stan przed modyfikacją, drugi prezentuje polecenie użyte do zmiany.
Jak potem zauważamy, rollback przywraca stan sprzed modyfikacji. Podczas
wystąpienia błędu w transakcji rollbackowana jest cała operacja
```

	PERSON_ID	FIRSTNAME	LASTNAME
1	1	Kornel	Makuszyński
2	2	Nikoła	Tesła
3	3	Albert	Einstein
4	4	Napoleon	Bonaparte
5	5	Robert	Oppenheimer
6	6	Issac	Newton
7	7	Niels	Bohr
8	8	Galileo	Galilei
9	9	Stephen	Hawking
10	10	Richard	Feynman

```
UPDATE PERSON
set FIRSTNAME = 'fakeName';
```

	PERSON_ID	FIRSTNAME	LASTNAME
1	1	fakeName	Makuszyński
2	2	fakeName	Tesła
3	3	fakeName	Einstein
4	4	fakeName	Bonaparte
5	5	fakeName	Oppenheimer
6	6	fakeName	Newton
7	7	fakeName	Bohr
8	8	fakeName	Galilei
9	9	fakeName	Hawking
10	10	fakeName	Feynman

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

```
-- SET TRANSACTION READ WRITE;

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, p.PERSON_ID
from RESERVATION r inner join PERSON p on p.PERSON_ID = r.PERSON_ID inner join
TRIP t on r.TRIP_ID = t.TRIP_ID;

commit

set TRANSACTION read write ;

create View vw_trip as
select outerT.TRIP_ID, outerT.COUNTRY, outerT.TRIP_DATE, outerT.TRIP_NAME,
outerT.MAX_NO_PLACES,
(MAX_NO_PLACES - (select COUNT (*) from RESERVATION
r inner join TRIP innerT on r.Trip_ID = innerT.Trip_ID where innerT.Trip_ID =
outerT.TRIP_ID))
as no_available_places from TRIP outerT

commit

set TRANSACTION read write;

create view vw_available_trip as
select MainT.TRIP_ID, MainT.COUNTRY, MainT.TRIP_DATE, MainT.TRIP_NAME,
MainT.MAX_NO_PLACES,
(MainT.MAX_NO_PLACES - (select COUNT (*) from RESERVATION
r inner join TRIP innerT on r.Trip_ID = innerT.Trip_ID where innerT.Trip_ID =
MainT.TRIP_ID))
```

```

as no_available_places from TRIP Maint
where (Maint.MAX_NO_PLACES - (select COUNT (*) from RESERVATION
r inner join TRIP innerT on r.Trip_ID = innerT.Trip_ID where innerT.Trip_ID =
Maint.Trip_ID)) > 0
and Maint.Trip_ID > '2024.03.19';

commit

```

	RESERVATION_ID	COUNTRY	TRIP_DATE	TRIP_NAME	FIRSTNAME	LASTNAME	STATUS	TRIP_ID	PERSON_ID
1	1	Grecja	2024-04-04	Wakacje All Inclusive	Kornel	Makuszyński	P	1	1
2	2	Grecja	2024-04-04	Wakacje All Inclusive	Nikola	Tesla	N	1	2
3	3	Poland	2024-06-24	Wakacje Wypaśnię	Kornel	Makuszyński	P	2	1
4	4	Germany	2024-12-15	Wakacje Pod Gruszą	Robert	Oppenheimer	N	3	5
5	5	Croatia	2024-03-17	Sea Holliday	Albert	Einstein	P	4	3
6	6	Poland	2024-06-24	Wakacje Wypaśnię	Niels	Bohr	P	2	7
7	7	Croatia	2024-03-17	Sea Holliday	Niels	Bohr	P	4	7
8	8	Poland	2024-06-24	Wakacje Wypaśnię	Galileo	Galilei	P	2	8
9	9	Germany	2024-12-15	Wakacje Pod Gruszą	Stephen	Hawking	P	3	9
10	10	Croatia	2024-03-17	Sea Holliday	Richard	Feynman	N	4	10

	TRIP_ID	COUNTRY	TRIP_DATE	TRIP_NAME	MAX_NO_PLACES	NO_AVAILABLE_PLACES
1	1	Grecja	2024-04-04	Wakacje All Inclusive	30	28
2	2	Poland	2024-06-24	Wakacje Wypaśnię	10	7
3	3	Germany	2024-12-15	Wakacje Pod Gruszą	13	11
4	4	Croatia	2024-03-17	Sea Holliday	20	17

	TRIP_ID	COUNTRY	TRIP_DATE	TRIP_NAME	MAX_NO_PLACES	NO_AVAILABLE_PLACES
1	1	Grecja	2024-04-04	Wakacje All Inclusive	30	28
2	2	Poland	2024-06-24	Wakacje Wypaśnię	10	7
3	3	Germany	2024-12-15	Wakacje Pod Gruszą	13	11

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_reservation**
- **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

```
-- set Transaction read write ;

create or REPLACE function f_trip_participants(SentTrip_ID NUMBER)
RETURN Varchar2
is
    participant_list VARCHAR2(100);
BEGIN
    select LISTAGG(P.FIRSTNAME || ' ' || P.LASTNAME, ', ') within group (order by
P.FIRSTNAME, P.LASTNAME)
    into participant_list
    from PERSON P
    inner join RESERVATION R on R.PERSON_ID = P.PERSON_ID
    inner join TRIP T on T.TRIP_ID = R.TRIP_ID
    where T.TRIP_ID = SentTrip_ID;

    RETURN participant_list;
end;
commit

select f_trip_participants(2) AS Trip_Participants from dual;

set TRANSACTION read write;
create or REPLACE FUNCTION f_person_reservation(SentPersonID NUMBER)
RETURN VARCHAR2
is
    ReservationList Varchar2(100);
BEGIN
    select LISTAGG(RESERVATION_ID, ', ') within group ( order by RESERVATION_ID)
    into ReservationList
    from RESERVATION R
    inner join PERSON P on P.PERSON_ID = R.PERSON_ID
    where P.PERSON_ID = SentPersonID;
```

```

RETURN ReservationList;
end;
commit

select f_person_reservation(2) as reservations from dual;

set transaction read write;
create or replace function f_available_trips_to(country_option IN VARCHAR2,
date_begin IN DATE, date_end IN DATE)
return VARCHAR2
is
    OptionList Varchar2(100);
BEGIN
    select LISTAGG(t.TRIP_NAME, ', ')
    into OptionList
    from TRIP t
    where t.COUNTRY = country_option and t.TRIP_DATE >= date_begin and t.TRIP_DATE
    <= date_end;

    return OptionList;
end;
commit

select f_available_trips_to('Grecja', TO_DATE('04.03.2023', 'DD.MM.YYYY'),
TO_DATE('05.07.2025', 'DD.MM.YYYY')) as test from dual;

--kontrola parametrów w funkcjach nie ma większego sensu gdyż jeśli parametr jest
--spoza zakresu to funkcja zwróci
--pustą listę a uważamy że jest to dobra odpowiedź na nieadekwatny parametr

```

TRIP_PARTICIPANTS	
1	Galileo Galilei, Kornel Makuszyński, Niels Bohr

RESERVATIONS	TEST
2	1 Wakacje All Inclusive

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 są 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

```
CREATE OR REPLACE PROCEDURE p_add_reservation (sentTripID IN Number, sentPersonID
IN Number)
IS
    v_trip_date DATE;
    v_max_places NUMBER;
    v_reserved_places NUMBER;
    v_reservationID NUMBER;
BEGIN

    SELECT TRIP_DATE, MAX_NO_PLACES
    INTO v_trip_date, v_max_places
```

```
FROM trip
WHERE TRIP_ID = sentTripID;

SELECT COUNT(*)
INTO v_reserved_places
FROM reservation
WHERE TRIP_ID = sentTripID;

SELECT COUNT(*) + 1
into v_reservationID
from RESERVATION;

IF v_trip_date < SYSDATE THEN
    RAISE_APPLICATION_ERROR(-20001, 'Cannot add reservation for a past
trip.');
```

trip.');

```
END IF;

IF v_reserved_places >= v_max_places THEN
    RAISE_APPLICATION_ERROR(-20002, 'No available places for this trip.');
```

trip.');

```
END IF;

INSERT INTO reservation (RESERVATION_ID, TRIP_ID, PERSON_ID, STATUS)
VALUES (v_reservationID, sentTripID, sentPersonID, 'P');
```

trip.');

```
INSERT INTO log (log_id, reservation_id, log_date, status)
VALUES (s_log_seq.nextval, v_reservationID, SYSDATE, 'P');
```

trip.');

```
COMMIT;
END;

commit;

CALL p_add_reservation(4, 3);

select * from RESERVATION

create or replace procedure p_modify_reservation_status(p_reservation_id IN
NUMBER, p_status IN VARCHAR2)
is
    committed_people_amount NUMBER;
    res_trip_id NUMBER;
    max_places_available NUMBER;
    curr_log_id NUMBER;
begin
    select TRIP_ID
    into res_trip_id
    from RESERVATION
```

```
where RESERVATION_ID = p_reservation_id;

SELECT COUNT(*)
INTO committed_people_amount
FROM RESERVATION
WHERE TRIP_ID = res_trip_id AND STATUS = 'P';

select MAX_NO_PLACES
into max_places_available
from TRIP
where TRIP_ID = res_trip_id;

if committed_people_amount >= max_places_available then
    RAISE_APPLICATION_ERROR(-20004, 'Cannot activate reservation. Trip is
already full.');
```

```
end if;

SELECT MAX(log_id) + 1 INTO curr_log_id FROM log;

update RESERVATION
set status = p_status
where RESERVATION_ID = p_reservation_id;





INSERT INTO log (LOG_ID, RESERVATION_ID, log_date, status)
values(curr_log_id, p_reservation_id, SYSDATE, p_status);





commit;

EXCEPTION
WHEN NO_DATA_FOUND THEN
    RAISE_APPLICATION_ERROR(-20001, 'Reservation not found.');
```

```
WHEN OTHERS THEN
    RAISE;
end;

CALL p_modify_reservation_status(1, 'N');
```

	 RESERVATION_ID ▾	 TRIP_ID ▾	 PERSON_ID ▾	 STATUS ▾
1	1	1	1	P
2	2	1	2	N
3	3	2	1	P
4	4	3	5	N
5	5	4	3	P
6	6	2	7	P
7	7	4	7	P
8	8	2	8	P
9	9	3	9	P
10	10	4	10	N
11	11	3	1	P

	 RESERVATION_ID ▾	 TRIP_ID ▾	 PERSON_ID ▾	 STATUS ▾
1	1	1	1	P
2	2	1	2	N
3	3	2	1	P
4	4	3	5	N
5	5	4	3	P
6	6	2	7	P
7	7	4	7	P
8	8	2	8	P
9	9	3	9	P
10	10	4	10	N
11	11	3	1	P
12	12	4	3	P

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 triggerów

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

```
-- create or replace trigger trg_add_reservation
after insert on RESERVATION
for each row
declare
    curr_id NUMBER;
begin
    select max(LOG_ID) + 1
    into curr_id
    from log;

    insert into log(log_id, reservation_id, log_date, status)
    values(curr_id, :new.RESERVATION_ID, SYSDATE, :new.STATUS);
end trg_add_reservation;

commit;

INSERT INTO RESERVATION (RESERVATION_ID, TRIP_ID, PERSON_ID, STATUS)
VALUES ((SELECT MAX(RESERVATION_ID) + 1 FROM RESERVATION), 3, 2, 'P');

commit;

create or replace trigger trg_modify_status
after update of STATUS on RESERVATION
for each row
declare
    curr_id NUMBER;
begin
    select max(log_ID) + 1
    into curr_id
    from LOG;

    insert into LOG(LOG_ID, RESERVATION_ID, LOG_DATE, STATUS)
    values(curr_id, :new.RESERVATION_ID, SYSDATE, :NEW.STATUS);

end;
```

```
commit;

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

```
END;

commit;
```

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 triggerów

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

```
-- create or replace trigger check_available_places
before insert on RESERVATION
for each row
declare
    max_places NUMBER;
```



```
        taken_places NUMBER;
begin
    select MAX_NO_PLACES
    into max_places
    from TRIP T
    where T.TRIP_ID = :new.TRIP_ID;

    select count (*)
    into taken_places
    from RESERVATION
    where TRIP_ID = :new.TRIP_ID and STATUS = 'P';

    if taken_places >= max_places then
        RAISE_APPLICATION_ERROR(-20001, 'Cannot add reservation. No available
places for this trip. ');
    end if;

end;
commit;

create or replace trigger check_status_availability
before update of status on RESERVATION
for each row
declare
    max_places NUMBER;
    taken_places NUMBER;
begin
    select MAX_NO_PLACES
    into max_places
    from TRIP T
    where T.TRIP_ID = :old.TRIP_ID;

    select count (*)
    into taken_places
    from RESERVATION
    where TRIP_ID = :old.TRIP_ID and STATUS = 'P';

    if taken_places >= max_places then
        RAISE_APPLICATION_ERROR(-20001, 'Cannot add reservation. No available
places for this trip. ');
    end if;

end;

commit;
```

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

```
-- ta procdura liczy wszystkie aktywne rezerwacje dla danego trip id i wynik jest
odejmowany
-- od maksymalnej wartosci dostepnych miejsc
create PROCEDURE update_no_available_places AS
BEGIN

  FOR r IN (SELECT TRIP_ID, COUNT(*) as liczba_rezerwacji FROM RESERVATION
    WHERE STATUS = 'P'
    GROUP BY TRIP_ID)
  LOOP
    UPDATE TRIP
      SET NO_AVAILABLE_PLACES = MAX_NO_PLACES - r.liczba_rezerwacji
      WHERE TRIP_ID = r.TRIP_ID;
  END LOOP;

  COMMIT;
END;
```

/

	TRIP_ID	TRIP_NAME	COUNTRY	TRIP_DATE	MAX_NO_PLACES	NO_AVAILABLE_PLACES
1	1	Wakacje All Inclusive	Grecja	2024-04-04	30	29
2	2	Wakacje Wypaśnie	Poland	2024-06-24	10	6
3	3	Wakacje Pod Gruszą	Germany	2024-12-15	13	10
4	4	Sea Holliday	Croatia	2024-03-17	20	18

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 triggerzy 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

```
-- procedura do dodawania rezerwacji praktycznie taka sama, tylko zmieniamy
--liczbe dostępnych miejsc o 1
```

```
create PROCEDURE p_add_reservation6a (sentTripID IN Number, sentPersonID IN
Number)
```

```
IS
```

```
    v_trip_date DATE;
    v_max_places NUMBER;
    v_reserved_places NUMBER;
    v_reservationID NUMBER;
```

```
BEGIN
```

```
    SELECT TRIP_DATE, MAX_NO_PLACES
    INTO v_trip_date, v_max_places
    FROM trip
    WHERE TRIP_ID = sentTripID;
```

```
SELECT COUNT(*)
INTO v_reserved_places
FROM reservation
WHERE TRIP_ID = sentTripID;

SELECT COUNT(*) + 1
into v_reservationID
from RESERVATION;

IF v_trip_date < SYSDATE THEN
    RAISE_APPLICATION_ERROR(-20001, 'Cannot add reservation for a past
trip.');
```

trip.');

```
END IF;

IF v_reserved_places >= v_max_places THEN
    RAISE_APPLICATION_ERROR(-20002, 'No available places for this trip.');
```

trip.');

```
END IF;

INSERT INTO reservation (RESERVATION_ID, TRIP_ID, PERSON_ID, STATUS)
VALUES (v_reservationID, sentTripID, sentPersonID, 'P');
```

trip.');

```
INSERT INTO log (log_id, reservation_id, log_date, status)
VALUES (s_log_seq.nextval, v_reservationID, SYSDATE, 'P');
```

trip.');

```
-- zmniejszenie no_avaliable_trips o 1
UPDATE trip
SET NO_AVAILABLE_PLACES = NO_AVAILABLE_PLACES - 1
WHERE TRIP_ID = sentTripID;

COMMIT;
END;
/

-- aby modyfikacja rezerwacji działała wyłączyliśmy 2 triggerry do tabeli
RESERVATIONS
BD_418001> ALTER TRIGGER TRG_MODIFY_STATUS DISABLE
[2024-03-25 22:50:10] completed in 20 ms
BD_418001> ALTER TRIGGER CHECK_STATUS_AVAILABILITY DISABLE
[2024-03-25 22:50:36] completed in 26 ms

-- Procedura korzysta z kolumny no_avaliable_places do odczytania dostępnych
miejsc
create PROCEDURE p_modify_reservation_status6a(
    p_reservation_id IN NUMBER,
    p_status IN VARCHAR2)
IS
```

```
previous_status VARCHAR2(1);
res_trip_id NUMBER;
available_places NUMBER;

BEGIN
    -- Pobranie TRIP_ID i poprzedniego statusu dla danej rezerwacji
    SELECT TRIP_ID, STATUS
    INTO res_trip_id, previous_status
    FROM RESERVATION
    WHERE RESERVATION_ID = p_reservation_id;

    -- Pobranie aktualnej liczby dostępnych miejsc
    SELECT NO_AVAILABLE_PLACES
    INTO available_places
    FROM TRIP
    WHERE TRIP_ID = res_trip_id;

    -- Logika zmiany liczby dostępnych miejsc
    IF previous_status = 'N' AND p_status = 'P' THEN
        -- Jeśli rezerwacja jest zmieniana z N na P, zmniejsz dostępne miejsca o 1
        IF available_places = 0 THEN
            RAISE_APPLICATION_ERROR(-20004, 'Cannot activate reservation. Trip is
already full.');
```

already full.');

```
        ELSE
            available_places := available_places - 1;
        END IF;
    ELSIF previous_status = 'P' AND p_status = 'N' THEN
        -- Jeśli rezerwacja jest zmieniana z P na N, zwiększ dostępne miejsca o 1
        available_places := available_places + 1;
    END IF;

    -- Aktualizacja liczby dostępnych miejsc w TRIP
    UPDATE TRIP
    SET NO_AVAILABLE_PLACES = available_places
    WHERE TRIP_ID = res_trip_id;

    -- Aktualizacja statusu rezerwacji
    UPDATE RESERVATION
    SET STATUS = p_status
    WHERE RESERVATION_ID = p_reservation_id;

    -- Logowanie zmiany
    INSERT INTO LOG (LOG_ID, RESERVATION_ID, LOG_DATE, STATUS)
    VALUES (s_log_seq.NEXTVAL, p_reservation_id, SYSDATE, p_status);

    COMMIT;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20001, 'Reservation or Trip not found.');
```

Reservation or Trip not found.');

```
    WHEN OTHERS THEN
        RAISE;
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

```
CREATE TRIGGER TRG_MODIFY_STATUS6b
AFTER UPDATE OF STATUS ON RESERVATION
FOR EACH ROW
BEGIN
    -- Jeśli status zmienia się z N na P, zmniejsz liczbę dostępnych miejsc
    IF :OLD.STATUS = 'N' AND :NEW.STATUS = 'P' THEN
        UPDATE TRIP
        SET NO_AVAILABLE_PLACES = NO_AVAILABLE_PLACES - 1
        WHERE TRIP_ID = :NEW.TRIP_ID
        AND NO_AVAILABLE_PLACES > 0;
    -- Jeśli status zmienia się z P na N, zwiększ liczbę dostępnych miejsc
    ELSIF :OLD.STATUS = 'P' AND :NEW.STATUS = 'N' THEN
        UPDATE TRIP
        SET NO_AVAILABLE_PLACES = NO_AVAILABLE_PLACES + 1
        WHERE TRIP_ID = :OLD.TRIP_ID;
    END IF;
END;
/

CREATE OR REPLACE TRIGGER TRG_ADD_RESERVATION6b
AFTER INSERT ON RESERVATION
FOR EACH ROW
BEGIN
```

```
-- Aktualizacja liczby dostępnych miejsc tylko dla potwierdzonych rezerwacji
IF :new.STATUS = 'P' THEN
    UPDATE TRIP
    SET NO_AVAILABLE_PLACES = NO_AVAILABLE_PLACES - 1
    WHERE TRIP_ID = :NEW.TRIP_ID
    AND NO_AVAILABLE_PLACES > 0;
END IF;
END;
/
```

Zadanie 7 - podsumowanie

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

```
-- na Podstawach Baz Danych na poprzednim semestrze używaliśmy MS SQL server
-- Ogólnie sposób robienia obu baz był dość podobny, tylko w samym pisaniu kodu
różnice
-- Np nie ma EXEC'a dla prodecury tylko trzeba zrobić BEGIN itp
-- Na początku było to lekko denerwujące, ale można się przywyzczać
-- Poza tym sposób myślenia jest praktycznie taki sam
```