

Laboratorium 1 - Karol Wrona

1. Tabele

```
create table person
(
  person_id int generated always as identity not null
, firstname varchar2(50)
, lastname varchar2(50)
, constraint person_pk primary key
(
  person_id
)
enable
);
```

```
create table trip
(
  trip_id int generated always as identity not null
, name varchar2(100)
, country varchar2(50)
, trip_date date
, max_no_places int
, constraint trip_pk primary key
(
  trip_id
)
enable
);
```

```
create table reservation
(
  reservation_id int generated always as identity not null
, trip_id int
, person_id int
, status char(1)
, no_places int
, constraint reservation_pk primary key
(
  reservation_id
)
enable
);
alter table reservation
add constraint reservation_fk1 foreign key
(
```

```
    person_id
  )
  references person
(
  person_id
)
enable;
alter table reservation
add constraint reservation_fk2 foreign key
(
  trip_id
)
references trip
(
  trip_id
)
enable;
alter table reservation
add constraint reservation_chk1 check
(status in ('n','p','c'))
enable;
```

Tabela log

Tabela dziennikująca zmiany w bazie danych. Dodano kolumnę info, która opisuje jaką operacja została wykonana. Triggery znajdują się w [punkcie 6](#).

```
create table log
(
  log_id int generated always as identity not null
, reservation_id int
, log_date date
, status char
, no_places int
, info varchar2(50)
, constraint log_pk primary key
(
  log_id
)
enable
);
alter table log
add constraint log_fk1 foreign key
(
  reservation_id
)
references RESERVATION
(
  reservation_id
)
enable;
```

2. Przykładowe dane

```
insert into person (firstname, lastname)
values('adam', 'kowalski');
insert into person (firstname, lastname)
values('jan', 'nowak');
select * from person;
insert into trip (name, country, trip_date, max_no_places)
values ('wycieczka do paryza', 'francja', to_date('2021-09-03', 'yyyy-mm-dd'), 3);
insert into trip (name, country, trip_date, max_no_places)
values ('wycieczka do krakowa', 'polska', to_date('2022-12-05', 'yyyy-mm-dd'), 2);
select * from trip;
insert into reservation(trip_id, person_id, no_places, status)
values (1, 1, 1, 'n');
insert into reservation(trip_id, person_id, no_places, status)
values (2, 1, 1, 'p');
```

3. Widoki

a)

```
create view reservations_view as
select t.country, t.trip_date, t.name,
       p.firstname, p.lastname, r.reservation_id,
       r.no_places, r.status
from reservation r
join trip t on r.trip_id = t.trip_id
join person p on r.person_id = p.person_id;
```

b) Korzystam z pomocniczej funkcji [available_places](#)

```
create view trips_view as
select t.country, t.trip_date, t.name,
       t.max_no_places, available_places(t.trip_id) as no_available_places
from trip t
join reservation r on r.reservation_id = t.trip_id;
```

c)

```
create view available_trips as
select t.country, t.trip_date, t.name,
       t.max_no_places, available_places(t.trip_id) as no_available_places
from trip t
```

```
join reservation r on r.reservation_id = t.trip_id
where available_places(t.trip_id) > 0;
```

Typy Danych

```
create or replace type reservation_type as OBJECT
(
  country varchar2(50),
  trip_date date,
  trip_name varchar2(50),
  firstname varchar2(50),
  lastname varchar2(50),
  reservation_id int,
  no_places int,
  status char(1)
);

create or replace type reservation_type_table is table of reservation_type;
```

```
create or replace type available_trip as OBJECT
(
  trip_id int,
  trip_name varchar2(50),
  country varchar2(50),
  trip_date date,
  available_places int
);

create or replace type available_trip_table is table of available_trip;
```

Funkcje Pomocnicze

```
create or replace function trip_exist(trip_id int)
return boolean
as
  exist number;
begin
  select count(r.trip_id) into exist
  from reservation r
  where r.trip_id = trip_exist.trip_id;

  if exist = 0 then
    return false;
  else
    return true;
end;
```

```
    end if;  
end;
```

```
create or replace function person_exist(person_id int)  
    return boolean  
as  
    exist number;  
begin  
    select count(p.person_id) into exist  
    from person p  
    where p.person_id = person_exist.person_id;  
  
    if exist = 0 then  
        return false;  
    else  
        return true;  
    end if;  
end;
```

```
create or replace function reservation_exist(reservation_id int)  
    return boolean  
as  
    exist number;  
begin  
    select count(r.reservation_id) into exist  
    from reservation r  
    where r.reservation_id = reservation_exist.reservation_id;  
  
    if exist = 0 then  
        return false;  
    else  
        return true;  
    end if;  
end;
```

```
create or replace function country_exist(country varchar2)  
    return boolean  
as  
    exist number;  
begin  
    select count(t.country) into exist  
    from trip t  
    where t.country = country_exist.country;  
  
    if exist = 0 then  
        return false;  
    else
```

```
        return true;
    end if;
end;
```

Funkcja obliczająca ile jest wolnych miejsc na danej wycieczce.

```
create or replace function available_places(trip_id int)
    return int
as
    result int;
begin
    select (t.max_no_places - sum(r.no_places)) as amount into result
    from trip t
    join reservation r on t.TRIP_ID = r.TRIP_ID
    where t.trip_id = available_places.trip_id and r.STATUS <> 'C'
    group by t.MAX_NO_PLACES;

    return result;
end;
```

4. Procedury pobierające dane

a)

```
create or replace function trip_participants(trip_id int)
    return reservation_type_table
as
    result reservation_type_table;
begin
    if not trip_exist(trip_id) then
        raise_application_error(-20000, 'Trip does not exist');
    end if;

    select reservation_type(t.country, t.trip_date, t.name,
        p.firstname, p.lastname, r.reservation_id,
        r.no_places, r.status)
    bulk collect
    into result
    from reservation r
    join trip t on r.trip_id = t.trip_id
    join person p on r.person_id = p.person_id
    where t.trip_id = trip_participants.trip_id;

    return result;
end;
```

b)

```
create or replace function person_reservations(person_id int)
    return reservation_type_table
as
    result reservation_type_table;
begin
    if not person_exist(person_id) then
        raise_application_error(-20000, 'Person does not exist');
    end if;

    select reservation_type(t.country, t.trip_date, t.name,
        p.firstname, p.lastname, r.reservation_id,
        r.no_places, r.status)
    bulk collect
    into result
    from person p
    join reservation r on p.person_id = r.person_id
    join trip t on t.trip_id = r.trip_id
    where p.person_id = person_reservations.person_id;

    return result;
end;
```

c)

```
create or replace function available_trips_to(country varchar2, date_from date,
date_to date)
    return available_trip_table
as
    result available_trip_table;
begin
    if not country_exist(country) then
        raise_application_error(-20000, 'There is no trip to this destination');
    ELSIF date_from > date_to then
        raise_application_error(-20000, 'Incorrect date');
    end if;

    select available_trip(t.trip_id, t.name, t.country, t.trip_date,
available_places(t.trip_id))
    bulk collect into result
    from trip t
    where t.country = available_trips_to.COUNTRY and
        t.trip_date between date_from and date_to;

    return result;
end;
```

5. Procedury modyfikujące dane

Część kontroli danych (np. czy jest wystarczająco dużo wolnych miejsc) została przeniesiona do [triggerów](#). W procedurach sprawdzamy tylko czy dane do których się odwołujemy istnieją.

a)

```
create or replace procedure add_reservation(trip_id int, person_id int, no_places
int)
as
    trip_start date;
begin
    select t.trip_date into trip_start
    from TRIP t where t.TRIP_ID = add_reservation.trip_id;

    if not trip_exist(trip_id) then
        RAISE_APPLICATION_ERROR(-20000, 'Trip does not exist');
    elsif not person_exist(person_id) then
        RAISE_APPLICATION_ERROR(-20000, 'Person does not exist');
    end if;

    insert into reservation(trip_id, person_id, status, no_places)
    values (trip_id, person_id, 'n', no_places);
end;
```

b)

```
create or replace procedure modify_reservation_status(reservation_id int, status
char)
as
    places int;
    trip_id int;
begin
    select r.no_places, r.trip_id into places, trip_id
    from reservation r
    where r.reservation_id = modify_reservation_status.reservation_id;

    if not reservation_exist(reservation_id) then
        raise_application_error(-20000, 'Reservation does not exist');
    end if;

    update reservation
    set status = modify_reservation_status.status
    where reservation_id = modify_reservation_status.reservation_id;
end;
```

c)

```
create or replace procedure modify_reservation(reservation_id int, no_places int)
as
```



```

trip_id int;
begin
  select r.trip_id into trip_id
  from reservation r
  where r.reservation_id = modify_reservation.reservation_id;

  if not reservation_exist(trip_id) then
    raise_application_error(-20000, 'Reservation does not exist');
  end if;

  update reservation
  set no_places = modify_reservation.no_places
  where reservation_id = modify_reservation.reservation_id;
end;
```

d)

```

create or replace procedure modify_max_places(trip_id int, no_places int)
as
  reserved_places int;
  current_max int;
begin
  select max_no_places into current_max
  from trip t
  where t.trip_id = modify_max_places.trip_id;

  reserved_places := current_max - available_places(trip_id);

  if not trip_exist(trip_id) then
    raise_application_error(-20000, 'Trip does not exist');
  end if;

  update trip
  set max_no_places = no_places
  where trip_id = modify_max_places.trip_id;
end;
```

6. Triggery dla tabeli log

Tabela log jest opisana w [punkcie 1](#).

a)

```

create or replace trigger log_add_reservation
after insert
on RESERVATION
for each row
begin
  insert into log (reservation_id, log_date, status, no_places, INFO)
```

```
values (:new.reservation_id, current_date, :new.status, :new.no_places, 'add
reservation');
end;
```

b) c)

Triggery opisane w treści zadania w podpunkcie b) i c) są zrealizowane jako jeden trigger gdyż oba wykonują update na tabeli reservation. W przypadku gdy wywołamy procedurę modify_reservation w taki sposób, że nie zmieni ona żadnych danych w tabeli, to taka operacja nie zostanie zapisana w dzienniku.

```
create or replace trigger log_modify_reservation
after update
on RESERVATION
for each row
begin
if :new.status <> :old.status then
insert into log (RESERVATION_ID, LOG_DATE, STATUS, NO_PLACES, INFO)
values (:new.reservation_id, current_date, :new.status, :new.no_places,
'changed status');
elsif :new.no_places <> :old.no_places then
insert into log (RESERVATION_ID, LOG_DATE, STATUS, NO_PLACES, INFO)
values (:new.reservation_id, current_date, :new.status, :new.no_places,
'changed no_places');
end if;
end;
```

7. Kontrola dostępności miejsc

a)

Rozsądnie byłoby zablokować możliwość zapisania się na wycieczkę, która już się odbyła. Jednak aby uniknąć problemów z sprawdzaniem poprawności innych procedur, opcja ta została zakomentowana (Być może Pan Dr. ma swój zestaw danych do przetestowania).

```
create or replace trigger reservation_add_trigger
before insert
on RESERVATION
for each row
declare
trip_start date;
begin
select t.trip_date into trip_start
from TRIP t where t.TRIP_ID = :new.trip_id;

if available_places(:new.trip_id) < :new.no_places then
RAISE_APPLICATION_ERROR(-20000, 'Not enough places');
elsif current_date > trip_start then
-- RAISE_APPLICATION_ERROR(-20000, 'The trip already started');
DBMS_OUTPUT.PUT_LINE('Uwaga wycieczka juz sie zaczęła');
```

```
    end if;  
end;
```

b)

```
create or replace trigger change_status_trigger  
    before insert  
    on RESERVATION  
    for each row  
declare  
    places int;  
    trip_id int;  
begin  
    select r.no_places, r.trip_id into places, trip_id  
    from reservation r  
    where r.reservation_id = :new.reservation_id;  
  
    if :new.status = 'c' and places > available_places(:new.trip_id) then  
        raise_application_error(-20000, 'Not enough available places');  
    elsif instr('cpn', :new.status) = 0 then  
        raise_application_error(-2000, 'Incorrect status');  
    end if;  
  
end;
```

c)

```
create or replace trigger modify_reservation_trigger  
    before insert  
    on RESERVATION  
    for each row  
declare  
    trip_id int;  
begin  
    select r.trip_id into trip_id  
    from reservation r  
    where r.reservation_id = :new.reservation_id;  
  
    if :new.no_places > available_places(trip_id) then  
        raise_application_error(-20000, 'Not enough places');  
    end if;  
  
end;
```

d)

```
create or replace trigger modify_max_places  
    before insert
```

```
on RESERVATION
for each row
declare
reserved_places int;
current_max int;
begin
select max_no_places into current_max
from trip t
where t.trip_id = :new.trip_id;

reserved_places := current_max - available_places(:new.trip_id);

if :new.no_places < reserved_places then
raise_application_error(-20000, 'New max_no_places is lower then ongoing
reservations');
end if;
end;
```