# LAB 1. Oracle PL/Sql

widoki, funkcje, procedury, triggery ćwiczenie
**Imiona i nazwiska autorów:** Tomasz Furgała, Konrad Tendaj, Łukasz Zegar

## Zadanie 0.

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:

`Transakcje` w systemach baz danych są niepodzielnymi jednostkami operacji, które zapewniają integralność danych poprzez atomowość, spójność, izolację i trwałość. Rozpoczynając, wykonując i zatwierdzając lub cofając grupę operacji, transakcje gwarantują, że baza danych pozostaje w spójnym stanie, nawet w przypadku awarii systemu. Ich działanie zapewnia niezawodność i integralność operacji na danych.

`Commit` Jeśli wszystkie operacje zostały wykonane poprawnie, transakcja jest zatwierdzana za pomocą polecenia `commit`, co powoduje trwałe zapisanie zmian w bazie danych.

`Rollback` Jeśli wystąpił błąd lub użytkownik zdecyduje się na anulowanie transakcji, można użyć polecenia `rollback`, aby cofnąć wszystkie zmiany wprowadzone w trakcie transakcji.

Można użyć bloków BEGIN ... EXCEPTION ... END do przechwytywania i obsługi błędów. W przypadku wystąpienia błędu, wykonanie transakcji może być przerwane, a transakcja może zostać automatycznie cofnięta (rollback), chyba że została zdefiniowana obsługa błędów, która określa, jak reagować na dany rodzaj błędu.

W Oracle PL/SQL i MS SQL Server T-SQL programowanie operacji wykorzystujących transakcje jest podobne pod względem koncepcji i celów. Oba języki umożliwiają rozpoczęcie, zatwierdzenie lub cofnięcie transakcji za pomocą poleceń BEGIN TRANSACTION, COMMIT i ROLLBACK. Jednak w Oracle PL/SQL często używa się bloków BEGIN ... EXCEPTION ... END do obsługi błędów, podczas gdy w T-SQL wykorzystuje się bloki BEGIN TRY ... END TRY; BEGIN CATCH ... END CATCH; dla tego samego celu

## Zadanie 1. 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

```
CREATE OR REPLACE VIEW VW_RESERVATION AS
SELECT R.RESERVATION_ID, T.COUNTRY, T.TRIP_DATE, T.TRIP_NAME,
       P.FIRSTNAME, P.LASTNAME, R.STATUS, R.TRIP_ID, R.PERSON_ID
FROM RESERVATION R
JOIN TRIP T ON R.TRIP_ID = T.TRIP_ID
JOIN PERSON P ON R.PERSON_ID = P.PERSON_ID;
```

| | RESERVATION_ID | COUNTRY | TRIP_DATE | TRIP_NAME | FIRSTNAME | LASTNAME | STATUS | TRIP_ID | PERSON_ID |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | Francja | 2023-09-12 | Wycieczka do Paryza | Jan | Nowak | P | 1 | 1 |
| 2 | 3 | Polska | 2025-05-03 | Piekny Krakow | Jan | Nowak | P | 2 | 1 |
| 3 | 2 | Francja | 2023-09-12 | Wycieczka do Paryza | Jan | Kowalski | N | 1 | 2 |
| 4 | 8 | Polska | 2025-05-01 | Hel | Jan | Kowalski | N | 4 | 2 |
| 5 | 6 | Francja | 2023-09-12 | Wycieczka do Paryza | Jan | Nowakowski | P | 1 | 3 |
| 6 | 9 | Polska | 2025-05-03 | Piekny Krakow | Jan | Nowakowski | C | 2 | 3 |
| 7 | 7 | Francja | 2023-09-12 | Wycieczka do Paryza | Novak | Nowak | C | 1 | 4 |
| 8 | 5 | Polska | 2025-05-03 | Piekny Krakow | Novak | Nowak | P | 2 | 4 |
| 9 | 4 | Polska | 2025-05-03 | Piekny Krakow | Novak | Nowak | C | 2 | 4 |
| 10 | 10 | Francja | 2025-05-01 | Znow do Francji | Anna | Nowak | P | 3 | 5 |
| 11 | 11 | Francja | 2025-05-01 | Znow do Francji | Piotr | Kowalczyk | N | 3 | 6 |
| 12 | 12 | Polska | 2025-05-01 | Hel | Monika | Wiśniewska | C | 4 | 7 |
| 13 | 13 | Polska | 2025-05-01 | Hel | Marcin | Lewandowski | P | 4 | 8 |

---

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)

Wykorzystuje złączenie lewe, aby uwzględnić wszystkie wycieczki, nawet te bez rezerwacji. Liczba dostępnych miejsc jest obliczana na podstawie różnicy między maksymalną liczbą miejsc a liczbą rezerwacji.

```
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 - COUNT(R.RESERVATION_ID) AS no_available_places
FROM TRIP T
LEFT JOIN RESERVATION R ON T.TRIP_ID = R.TRIP_ID AND R.STATUS != 'C'
GROUP BY T.TRIP_ID, T.COUNTRY, T.TRIP_DATE, T.TRIP_NAME, T.MAX_NO_PLACES
```

| | TRIP_ID | COUNTRY | TRIP_DATE | TRIP_NAME | MAX_NO_PLACES | NO_AVAILABLE_PLACES |
|---|---|---|---|---|---|---|
| 1 | 3 | Francja | 2025-05-01 | Znow do Francji | 2 | 0 |
| 2 | 1 | Francja | 2023-09-12 | Wycieczka do Paryza | 3 | 0 |
| 3 | 2 | Polska | 2025-05-03 | Piekny Krakow | 4 | 2 |
| 4 | 4 | Polska | 2025-05-01 | Hel | 2 | 0 |

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)

Korzystamy z poprzeniego widoku ( VW_TRIP)

```
CREATE OR REPLACE VIEW VW_AVAILABLE_TRIP AS
SELECT * FROM VW_TRIP
WHERE TRIP_DATE > SYSDATE AND NO_AVAILABLE_PLACES > 0;
```

| | TRIP_ID | COUNTRY | TRIP_DATE | TRIP_NAME | MAX_NO_PLACES | NO_AVAILABLE_PLACES |
|---|---|---|---|---|---|---|
| 1 | 2 | Polska | 2025-05-03 | Piekny Krakow | 4 | 2 |

# Zadanie 2. Funkcje

f_trip_participants zadaniem funkcji jest zwrócenie listy uczestników wskazanej wycieczki parametry funkcji: trip_id funkcja zwraca podobny zestaw danych jak widok vw_eservation

```
CREATE OR REPLACE FUNCTION f_trip_participants (p_trip_id NUMBER)
    RETURN SYS_REFCURSOR
```

```
IS
    participants_cursor SYS_REFCURSOR;
    trip_exists NUMBER;
BEGIN
    IF p_trip_id IS NULL THEN
        RAISE_APPLICATION_ERROR(-20001, 'Parametry funkcji muszą być określone.');
    END IF;

    SELECT COUNT(*) INTO trip_exists FROM TRIP WHERE trip_id = p_trip_id;

    IF trip_exists = 0 THEN
        RAISE_APPLICATION_ERROR(-20002, 'Wycieczka o podanym trip_id nie
istnieje.');
    END IF;

    OPEN participants_cursor FOR
        SELECT * FROM vw_reservation WHERE trip_id = p_trip_id;
    RETURN participants_cursor;
END;
```

f_person_reservations zadaniem funkcji jest zwrócenie listy rezerwacji danej osoby parametry funkcji: person_id funkcja zwraca podobny zestaw danych jak widok vw_reservation

```
CREATE or REPLACE FUNCTION f_person_reservations (p_person_id NUMBER)
    RETURN SYS_REFCURSOR
IS
    participants_cursor SYS_REFCURSOR;
    person_exists NUMBER;
BEGIN
    IF p_person_id IS NULL THEN
        RAISE_APPLICATION_ERROR(-20001, 'Parametry funkcji muszą być określone.');
    END IF;

    SELECT COUNT(*) INTO person_exists FROM PERSON WHERE PERSON_ID = p_person_id;

    IF person_exists = 0 THEN
        RAISE_APPLICATION_ERROR(-20002, 'Wycieczka o podanym trip_id nie
istnieje.');
    END IF;

    OPEN participants_cursor FOR
        SELECT * FROM vw_reservation WHERE person_id = p_person_id;
    RETURN participants_cursor;
end;
```

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

```
CREATE or REPLACE FUNCTION f_available_trips_to (p_country VARCHAR(50),
p_date_from date, p_date_to date)
    RETURN SYS_REFCURSOR
IS
    participants_cursor SYS_REFCURSOR;
BEGIN
    IF p_country IS NULL OR p_date_from IS NULL OR p_date_to IS NULL THEN
        RAISE_APPLICATION_ERROR(-20001, 'Wszystkie parametry funkcji muszą być
określone.');
    END IF;

    IF p_date_from > p_date_to THEN
        RAISE_APPLICATION_ERROR(-20002, 'Data początkowa nie może być większa niż
data końcowa.');
    END IF;


    OPEN participants_cursor FOR
        SELECT * FROM TRIP WHERE COUNTRY = p_country AND TRIP_DATE > p_date_from
AND TRIP_DATE < p_date_to;
    RETURN participants_cursor;
end;
```

Czy kontrola parametrów w przypadku funkcji ma sens?

- jakie są zalety/wady takiego rozwiązania?

Zalety:

1. Bezpieczeństwo: Kontrola parametrów pozwala zapobiec wykonaniu funkcji z nieprawidłowymi danymi lub unieważnionymi wartościami parametrów, co może prowadzić do błędów lub niewłaściwych wyników.
2. Ochrona danych: Poprawna kontrola parametrów może chronić dane w bazie przed błędnym dostępem lub zmianami spowodowanymi przez nieprawidłowe parametry.
3. Poprawność danych: Zapewnienie, że funkcja otrzymuje poprawne dane, pomaga zachować integralność danych w systemie.

Wady:

1. Złożoność kodu: Dodanie kontroli parametrów może zwiększyć złożoność kodu funkcji, co może prowadzić do większej liczby linii kodu i trudniejszej analizy i debugowania.
2. Wydajność: Długotrwałe lub złożone kontrole parametrów mogą wpływać na wydajność funkcji, zwłaszcza gdy są wywoływane wielokrotnie w złożonych zapytaniach.

## Zadanie 3. Procedury

- pomocnicza funkcja **f_get_reserved_places** - zwraca liczbę zarezerwowanych miejsc na daną wycieczkę podając jej id

```
create function f_get_reserved_places(a_trip_id int)
return int
is
    s_reserved_places int;
begin
    select count(RESERVATION_ID) into s_reserved_places
    from RESERVATION r
    where r.TRIP_ID = a_trip_id and r.status in ('N', 'P');

    return s_reserved_places;
end;
```

---

- pomocnicza funkcja **f_check_free_places** - zwraca liczbę wolnych miejsc na daną wycieczkę podając jej id

```
create function f_check_free_places(a_trip_id int)
return int
is
    s_free_places int;
begin
    select MAX_NO_PLACES - f_get_reserved_places(a_trip_id)
    into s_free_places
    from TRIP
    where TRIP_ID = a_trip_id;
    return s_free_places;
end;
```

---

- **p_add_reservation** - dopisanie nowej rezerwacji

```
create procedure p_add_reservation(a_trip_id int, a_person_id int)
as
    s_trip_date date;
    s_count int;
    s_reservation_id int;
begin
    -- Sprawdzenie czy osoba istnieje w systemie
    select count(*) into s_count from PERSON where PERSON_ID = a_person_id;
    if s_count = 0 then RAISE_APPLICATION_ERROR(-20001, 'Nie istnieje osoba o
podanym ID.');
    end if;

    -- Sprawdzenie czy wycieczka istnieje w systemie
    select count(*) into s_count from TRIP where TRIP_ID = a_trip_id;
    if s_count = 0 then RAISE_APPLICATION_ERROR(-20002, 'Wycieczka o podanym ID
nie istnieje.');
    end if;
```

```
    -- Sprawdzenie czy wycieczka się już nie odbyła
    select TRIP_DATE into s_trip_date from TRIP t where t.TRIP_ID = a_trip_id;
    DBMS_OUTPUT.PUT_LINE('Data wycieczki: ' || s_trip_date);
    if s_trip_date < SYSDATE then RAISE_APPLICATION_ERROR(-20003, 'Wycieczka już
się odbyła.');
    end if;

    -- Sprawdzenie czy osoba jest już zapisana na tą wycieczkę
    select COUNT(*) into s_count from RESERVATION
    where TRIP_ID = a_trip_id and PERSON_ID = a_person_id;
    if s_count > 0 then RAISE_APPLICATION_ERROR(-20004, 'Ta osoba jest już
zapisana na tę wycieczkę.');
    end if;

    -- Sprawdzenie czy jest wolne miejsce
    if f_check_free_places(a_trip_id) <= 0 then
        RAISE_APPLICATION_ERROR(-20005, 'Brak wolnych miejsc na wycieczce.');
    end if;

    -- Wstawienie rekordu do RESERVATION i pobranie RESERVATION_ID
    insert into RESERVATION (TRIP_ID, PERSON_ID, STATUS)
    values (a_trip_id, a_person_id, 'N')
    returning RESERVATION_ID into s_reservation_id;

    -- Wstawienie rekordu do LOG
    insert into LOG (RESERVATION_ID, LOG_DATE, STATUS)
    values (s_reservation_id, SYSDATE, 'N');

    DBMS_OUTPUT.PUT_LINE('Rezerwacja została pomyślnie dodana.');
exception
    when others then rollback;
        DBMS_OUTPUT.PUT_LINE('Wystąpił błąd: ' || SQLERRM);
end;
```

**Procedura posiada wiele zabezpieczeń jak np.:**

```
BD_416125> begin
               p_add_reservation(5, 1);
           end;
[2024-03-29 12:28:26] completed in 55 ms
Wystąpił błąd: ORA-20002: Wycieczka o podanym ID nie istnieje.
```

```
BD_416125> begin
               p_add_reservation(3, 5);
           end;
[2024-03-29 12:33:17] completed in 31 ms
Data wycieczki: 25/05/01
Wystąpił błąd: ORA-20004: Ta osoba jest już zapisana na tę wycieczkę.
```

```
BD_416125> begin
               P_ADD_RESERVATION(3, 3);
           end;
[2024-03-29 12:47:47] completed in 23 ms
Data wycieczki: 25/05/01
Wystąpił błąd: ORA-20005: Brak wolnych miejsc na wycieczce.
```

**Jednakże jeśli nie ma żadnych problemów to:**

```
BD_416125> begin
               P_ADD_RESERVATION(5, 3);
           end;
[2024-03-29 12:51:49] completed in 85 ms
Data wycieczki: 24/04/04
Rezerwacja została pomyślnie dodana.
```

---

- **p_modify_reservation_status** - zmiana statusu rezerwacji

```
create procedure p_modify_reservation_status(a_reservation_id int, a_status char)
as
    s_trip_id int;
    s_curr_status char;
    s_trip_date date;
begin
    -- Poprawność argumentu a_status
    if a_status not in ('N', 'P', 'C') then
        RAISE_APPLICATION_ERROR(-20001, 'Niepoprawny status.');
    end if;

    -- Pobranie TRIP_ID i STATUS
    select TRIP_ID, STATUS into s_trip_id, s_curr_status
    from RESERVATION where RESERVATION_ID = a_reservation_id;

    -- Sprawdzenie istnienia rezerwacji
    if s_trip_id is null then
        RAISE_APPLICATION_ERROR(-20002, 'Rezerwacja o podanym ID nie istnieje.');
    end if;

    -- Sprawdzenie czy rezerwacja nie ma już podanego statusu
    if s_curr_status = a_status then
        RAISE_APPLICATION_ERROR(-20001, 'Rezerwacja ma już ten status.');
    end if;

    -- Data wycieczki
    select TRIP_DATE into s_trip_date from TRIP
    where TRIP_ID = s_trip_id;

    if s_curr_status = 'P' AND a_status = 'N' then
        RAISE_APPLICATION_ERROR(-20006, 'Nie można zmienić statusu na N, gdy jest
```

```
już opłacona rezerwacja.');
    end if;

    if a_status = 'C' then -- anulowanie rezerwacji
        -- Sprawdzenie czy wycieczka się już nie odbyła
        if s_trip_date < SYSDATE then
            RAISE_APPLICATION_ERROR(-20003, 'Wycieczka już się odbyła. Nie można
zmienić statusu');
        end if;

        update RESERVATION set STATUS = a_status where RESERVATION_ID =
a_reservation_id;
        insert into LOG (RESERVATION_ID, LOG_DATE, STATUS) values
(a_reservation_id, SYSDATE, a_status);
        DBMS_OUTPUT.PUT_LINE('Twoja rezerwacja została anulowana.');

        return;
    end if;

    if s_curr_status = 'C' then
        -- Sprawdzenie czy jest wolne miejsce
        if f_check_free_places(s_trip_id) <= 0 then
            RAISE_APPLICATION_ERROR(-20005, 'Brak wolnych miejsc na wycieczce.');
        end if;

        update RESERVATION set STATUS = a_status where RESERVATION_ID =
a_reservation_id;
        insert into LOG (RESERVATION_ID, LOG_DATE, STATUS) values
(a_reservation_id, SYSDATE, a_status);
        DBMS_OUTPUT.PUT_LINE('Status twojej rezerwacji został zaaktualizowany.');
        return;
    end if;

    if a_status = 'P' then
        update RESERVATION set STATUS = a_status where RESERVATION_ID =
a_reservation_id;
        insert into LOG (RESERVATION_ID, LOG_DATE, STATUS) values
(a_reservation_id, SYSDATE, a_status);
        DBMS_OUTPUT.PUT_LINE('Twoje miejsce jest już potwierdzone i zapłacone.');
    end if;
end;
```

**Przykładowe zabezpieczenia:**

```
BD_416125> begin
               P_MODIFY_RESERVATION_STATUS(4, 'C');
           end;
[2024-03-29 18:49:03] [72000][20001]
[2024-03-29 18:49:03]   ORA-20001: Rezerwacja ma już ten status.
```

```
BD_416125> begin
               P_MODIFY_RESERVATION_STATUS(4, 'Wrong');
           end;
[2024-03-29 18:53:43] [72000][20001]
[2024-03-29 18:53:43]   ORA-20001: Niepoprawny status.
```

```
BD_416125> begin
               P_MODIFY_RESERVATION_STATUS(12, 'N');
           end;
[2024-03-29 18:58:06] [72000][20005]
[2024-03-29 18:58:06]   ORA-20005: Brak wolnych miejsc na wycieczce.
```

**Oczywiście procedura działa:**

```
BD_416125> begin
               P_MODIFY_RESERVATION_STATUS(14, 'P');
           end;
[2024-03-29 19:00:18] completed in 23 ms
```

- **p_modify_max_no_places** - zmiana maksymalnej liczby miejsc na daną wycieczkę

```
create procedure p_modify_max_no_places(a_trip_id int, a_max_no_places int)
as
    s_count int;
begin
    if a_max_no_places <= 0 then
        RAISE_APPLICATION_ERROR(-20001, 'Proszę podać poprawną liczbę');
    end if;

    select count(*) into s_count from TRIP WHERE TRIP_ID = a_trip_id;
    if s_count = 0 then
        RAISE_APPLICATION_ERROR(-20002, 'Nie istnieje wycieczka o podanym ID');
    end if;

    if f_get_reserved_places(a_trip_id) > a_max_no_places then
        RAISE_APPLICATION_ERROR(-20003, 'Nie można zmiejszyć liczby miejsc poniżej
liczby zarezerwowanych');
    end if;

    update TRIP
    set MAX_NO_PLACES = a_max_no_places where TRIP_ID = a_trip_id;
exception
    when others then rollback;
        raise;
end;
```

---

# Zadanie 4. Triggery

- **trg_reservation_log** - tworzy log'a przy dodaniu / aktualizacji statusu rezerwacji

```
create or replace trigger trg_reservation_log
after insert or update on RESERVATION
for each row
begin
    if :OLD.STATUS != :NEW.STATUS then
        insert into LOG (RESERVATION_ID, LOG_DATE, STATUS)
        values (:NEW.RESERVATION_ID, SYSDATE, :NEW.STATUS);
    end if;
end;
```

---

- **trg_prevent_reservation_deletion** - blokuje możliwość usuwania rezerwacji

```
create or replace trigger trg_prevent_reservation_deletion
before delete on RESERVATION
for each row
begin
    RAISE_APPLICATION_ERROR(-20001, 'Usunięcie rezerwacji jest zabronione.');
end;
```

```
BD_416125> delete
           from RESERVATION
           where RESERVATION_ID = 1
[2024-03-29 19:13:48] [72000][20001]
[2024-03-29 19:13:48]   ORA-20001: Usunięcie rezerwacji jest zabronione.
```

- **Aktualizacja procedur** polega na usunięciu wierszów z dodawaniem do log'ów

**Z procedury p_add_reservation usuwamy zmienną *s_reservation_id* oraz zmieniamy:**

```
    -- Wstawienie rekordu do RESERVATION i pobranie RESERVATION_ID
    insert into RESERVATION (TRIP_ID, PERSON_ID, STATUS)
    values (a_trip_id, a_person_id, 'N')
    returning RESERVATION_ID into s_reservation_id;

    -- Wstawienie rekordu do LOG
    insert into LOG (RESERVATION_ID, LOG_DATE, STATUS)
    values (s_reservation_id, SYSDATE, 'N');
```

**na:**

```
  -- Wstawienie rekordu do RESERVATION i pobranie RESERVATION_ID
  insert into RESERVATION (TRIP_ID, PERSON_ID, STATUS)
  values (a_trip_id, a_person_id, 'N');
```

```
create or replace procedure p_add_reservation_4(a_trip_id int, a_person_id int)
as
    s_trip_date date;
    s_count int;
    s_reservation_id int;
begin
  -- Sprawdzenie czy osoba istnieje w systemie
    select count(*) into s_count from PERSON where PERSON_ID = a_person_id;
    if s_count = 0 then RAISE_APPLICATION_ERROR(-20001, 'Nie istnieje osoba o
podanym ID.');
    end if;

    -- Sprawdzenie czy wycieczka istnieje w systemie
    select count(*) into s_count from TRIP where TRIP_ID = a_trip_id;
    if s_count = 0 then RAISE_APPLICATION_ERROR(-20002, 'Wycieczka o podanym ID
nie istnieje.');
    end if;

    -- Sprawdzenie czy wycieczka się już nie odbyła
    select TRIP_DATE into s_trip_date from TRIP t where t.TRIP_ID = a_trip_id;
    DBMS_OUTPUT.PUT_LINE('Data wycieczki: ' || s_trip_date);
    if s_trip_date < SYSDATE then RAISE_APPLICATION_ERROR(-20003, 'Wycieczka już
się odbyła.');
    end if;

    -- Sprawdzenie czy osoba jest już zapisana na tą wycieczkę
    select COUNT(*) into s_count from RESERVATION
    where TRIP_ID = a_trip_id and PERSON_ID = a_person_id;
    if s_count > 0 then RAISE_APPLICATION_ERROR(-20004, 'Ta osoba jest już
zapisana na tę wycieczkę.');
    end if;

    -- Sprawdzenie czy jest wolne miejsce
    if f_check_free_places(a_trip_id) <= 0 then
        RAISE_APPLICATION_ERROR(-20005, 'Brak wolnych miejsc na wycieczce.');
    end if;

    -- Wstawienie rekordu do RESERVATION i pobranie RESERVATION_ID
    insert into RESERVATION (TRIP_ID, PERSON_ID, STATUS)
    values (a_trip_id, a_person_id, 'N');

    DBMS_OUTPUT.PUT_LINE('Rezerwacja została pomyślnie dodana.');
exception
 when others then rollback;
        DBMS_OUTPUT.PUT_LINE('Wystąpił błąd: ' || SQLERRM);
end;
```

**Natomiast z procedury p_modify_reservation_status usuwamy 3 takie wiersze:**

```
insert into LOG (RESERVATION_ID, LOG_DATE, STATUS) values (a_reservation_id,
SYSDATE, a_status);
```

```
create or replace procedure p_modify_reservation_status_4(a_reservation_id int,
a_status char)
as
    s_trip_id int;
    s_curr_status char;
    s_trip_date date;
begin
    -- Poprawność argumentu a_status
    if a_status not in ('N', 'P', 'C') then
        RAISE_APPLICATION_ERROR(-20001, 'Niepoprawny status.');
    end if;


    -- Pobranie TRIP_ID i STATUS
    select TRIP_ID, STATUS into s_trip_id, s_curr_status
    from RESERVATION where RESERVATION_ID = a_reservation_id;

    -- Sprawdzenie istnienia rezerwacji
    if s_trip_id is null then
        RAISE_APPLICATION_ERROR(-20002, 'Rezerwacja o podanym ID nie istnieje.');
    end if;

    -- Sprawdzenie czy rezerwacja nie ma już podanego statusu
    if s_curr_status = a_status then
        RAISE_APPLICATION_ERROR(-20001, 'Rezerwacja ma już ten status.');
    end if;

  -- Data wycieczki
  select TRIP_DATE into s_trip_date from TRIP
        where TRIP_ID = s_trip_id;

  if s_curr_status = 'P' AND a_status = 'N' then
      RAISE_APPLICATION_ERROR(-20006, 'Nie można zmienić statusu na N, gdy jest
już opłacona rezerwacja.');
  end if;



  if a_status = 'C' then -- anulowanie rezerwacji
      -- Sprawdzenie czy wycieczka się już nie odbyła
      if s_trip_date < SYSDATE then
            RAISE_APPLICATION_ERROR(-20003, 'Wycieczka już się odbyła. Nie można
zmienić statusu');
      end if;
```

```
        update RESERVATION set STATUS = a_status where RESERVATION_ID =
    a_reservation_id;
        DBMS_OUTPUT.PUT_LINE('Twoja rezerwacja została anulowana.');

        return;
      end if;

      if s_curr_status = 'C' then
        -- Sprawdzenie czy jest wolne miejsce
        if f_check_free_places(s_trip_id) <= 0 then
            RAISE_APPLICATION_ERROR(-20005, 'Brak wolnych miejsc na wycieczce.');
        end if;

        update RESERVATION set STATUS = a_status where RESERVATION_ID =
    a_reservation_id;
        DBMS_OUTPUT.PUT_LINE('Status twojej rezerwacji został zaaktualizowany.');
        return;
      end if;

      if a_status = 'P' then
        update RESERVATION set STATUS = a_status where RESERVATION_ID =
    a_reservation_id;
        DBMS_OUTPUT.PUT_LINE('Twoje miejsce jest już potwierdzone i zapłacone.');
      end if;
    end;
```

---

## Zadanie 5. Triggery

- **trg_insert_reservation** - sprawdza czy jest wolne miejsce do nowej rezerwacji

```
create or replace trigger trg_insert_reservation
before insert on RESERVATION
for each row
begin
    if f_check_free_places(:NEW.TRIP_ID) <= 0 then
        RAISE_APPLICATION_ERROR(-20005, 'Brak wolnych miejsc na wycieczce.');
        ROLLBACK;
    end if;
end;
```

```
BD_416125> begin
               P_ADD_RESERVATION_5(3, 7);
           end;
[2024-03-29 19:33:46] completed in 40 ms
Data wycieczki: 25/05/01
Wystąpił błąd: ORA-20005: Brak wolnych miejsc na wycieczce.
```

- **trg_update_reservation_status** - sprawdza czy jest wolne miejsce do odnowienia rezerwacji ze statusu anulowanej 'C' na nową 'N' / zapłaconą 'P'

```
create or replace trigger trg_update_reservation_status
before update on RESERVATION
for each row
DECLARE
    v_trip_date DATE;
begin
    if :OLD.STATUS = 'C' and f_check_free_places(:NEW.TRIP_ID) <= 0 then
        RAISE_APPLICATION_ERROR(-20005, 'Brak wolnych miejsc na wycieczce.');
        Rollback;
    end if;

    if :OLD.STATUS = 'P' AND :NEW.STATUS = 'N' then
        RAISE_APPLICATION_ERROR(-20006, 'Nie można zmienić statusu na N, gdy jest
już opłacona rezerwacja.');
        Rollback;
    end if;

    SELECT trip_date INTO v_trip_date
    FROM TRIP
    WHERE TRIP_ID = :OLD.TRIP_ID;
    if v_trip_date < SYSDATE then
        RAISE_APPLICATION_ERROR(-20003, 'Wycieczka już się odbyła. Nie można
zmienić statusu');
        Rollback;
  end if;
end;
```

```
BD_416125> begin
             P_MODIFY_RESERVATION_STATUS(3, 'N');
           end;
[2024-03-29 22:00:46] [72000][20006]
[2024-03-29 22:00:46]   ORA-20006: Nie można zmienić statusu na N, gdy jest już opłacona rezerwacja.
```

## Aktualizacja procedur

**Z procedury p_add_reservation usuwamy ten fragment:**

```
    -- Sprawdzenie czy jest wolne miejsce
    if f_check_free_places(a_trip_id) <= 0 then
      RAISE_APPLICATION_ERROR(-20005, 'Brak wolnych miejsc na wycieczce.');
    end if;
```

```
    create or replace procedure p_add_reservation_5(a_trip_id int, a_person_id int)
    as
      s_trip_date date;
```

```
      s_count int;
  begin
    -- Sprawdzenie czy osoba istnieje w systemie
    select count(*) into s_count from PERSON where PERSON_ID = a_person_id;
      if s_count = 0 then RAISE_APPLICATION_ERROR(-20001, 'Nie istnieje osoba o
podanym ID.');
      end if;

      -- Sprawdzenie czy wycieczka istnieje w systemie
    select count(*) into s_count from TRIP where TRIP_ID = a_trip_id;
      if s_count = 0 then RAISE_APPLICATION_ERROR(-20002, 'Wycieczka o podanym ID
nie istnieje.');
      end if;

      -- Sprawdzenie czy wycieczka się już nie odbyła
    select TRIP_DATE into s_trip_date from TRIP t where t.TRIP_ID = a_trip_id;
      DBMS_OUTPUT.PUT_LINE('Data wycieczki: ' || s_trip_date);
      if s_trip_date < SYSDATE then RAISE_APPLICATION_ERROR(-20003, 'Wycieczka już
się odbyła.');
      end if;

      -- Sprawdzenie czy osoba jest już zapisana na tą wycieczkę
    select COUNT(*) into s_count from RESERVATION
      where TRIP_ID = a_trip_id and PERSON_ID = a_person_id;
      if s_count > 0 then RAISE_APPLICATION_ERROR(-20004, 'Ta osoba jest już
zapisana na tę wycieczkę.');
      end if;

    -- Wstawienie rekordu do RESERVATION i pobranie RESERVATION_ID
    insert into RESERVATION (TRIP_ID, PERSON_ID, STATUS)
    values (a_trip_id, a_person_id, 'N');

      DBMS_OUTPUT.PUT_LINE('Rezerwacja została pomyślnie dodana.');
  exception
  when others then rollback;
          DBMS_OUTPUT.PUT_LINE('Wystąpił błąd: ' || SQLERRM);
  end;
```

**Procedura "p_modify_reservation_status_5":**

```
create or replace procedure p_modify_reservation_status_5(a_reservation_id int,
a_status char)
as
  s_trip_id int;
  s_curr_status char;
begin
  -- Poprawność argumentu a_status
  if a_status not in ('N', 'P', 'C') then
  RAISE_APPLICATION_ERROR(-20001, 'Niepoprawny status.');
    end if;
```

```
    -- Pobranie TRIP_ID i STATUS
  select TRIP_ID, STATUS into s_trip_id, s_curr_status
    from RESERVATION where RESERVATION_ID = a_reservation_id;

    -- Sprawdzenie istnienia rezerwacji
  if s_trip_id is null then
  RAISE_APPLICATION_ERROR(-20002, 'Rezerwacja o podanym ID nie istnieje.');
    end if;

    -- Sprawdzenie czy rezerwacja nie ma już podanego statusu
  if s_curr_status = a_status then
  RAISE_APPLICATION_ERROR(-20001, 'Rezerwacja ma już ten status.');
    end if;

  if a_status = 'C' then -- anulowanie rezerwacji
    update RESERVATION set STATUS = a_status where RESERVATION_ID =
a_reservation_id;
    DBMS_OUTPUT.PUT_LINE('Twoja rezerwacja została anulowana.');
    return;
  end if;

  if s_curr_status = 'C' then
    update RESERVATION set STATUS = a_status where RESERVATION_ID =
a_reservation_id;
    DBMS_OUTPUT.PUT_LINE('Status twojej rezerwacji został zaaktualizowany.');
    return;
  end if;


  if a_status = 'P' then
    update RESERVATION set STATUS = a_status where RESERVATION_ID =
a_reservation_id;
    DBMS_OUTPUT.PUT_LINE('Twoje miejsce jest już potwierdzone i zapłacone.');
  end if;
end;
```

---

# Zadanie 6.

**Polecenie dodające nową kolumne do `Trip`**

```
ALTER TABLE trip
ADD no_available_places INT;
```

**Procedura do aktualizowania pola `no_available_places` dla danego `trip_id` i wywołanie jej dla kolejnych wierszy**

```
CREATE OR REPLACE PROCEDURE p_update_no_available_places(a_trip_id VARCHAR) AS
BEGIN
    UPDATE TRIP
    SET no_available_places =
        MAX_NO_PLACES - (
            SELECT COUNT(*)
            FROM reservation
            WHERE trip_id = a_trip_id
            AND status IN ('P', 'N')
        )
    WHERE trip_id = a_trip_id;
END p_update_no_available_places;


  DECLARE
      CURSOR c_trips IS
          SELECT trip_id FROM trip;
  BEGIN
      FOR trip_rec IN c_trips LOOP
          p_update_no_available_places(trip_rec.trip_id);
      END LOOP;
  END;
```

**Uproszczenie widoków:**

```
CREATE OR REPLACE VIEW VW_TRIP_6 AS
SELECT T.TRIP_ID, T.COUNTRY, T.TRIP_DATE, T.TRIP_NAME, T.MAX_NO_PLACES,
T.NO_AVAILABLE_PLACES
FROM TRIP T;
```

## Zadanie 6a.

**Na koniec każdej procedury wywołujemy procedure `P_UPDATE_NO_AVAILABLE_PLACES(a_trip_id);` aktualizującą liczbę wolnych miejsc.**

**Modyfikacja procedury p_add_reservation:**

```
create or replace procedure p_add_reservation_6a(a_trip_id int, a_person_id int)
as
  s_trip_date date;
  s_count int;
  s_reservation_id int;
  s_available_places int;
begin
  -- Sprawdzenie czy osoba istnieje w systemie
  select count(*) into s_count from PERSON where PERSON_ID = a_person_id;
    if s_count = 0 then RAISE_APPLICATION_ERROR(-20001, 'Nie istnieje osoba o
```

```
podanym ID.');
    end if;

    -- Sprawdzenie czy wycieczka istnieje w systemie
  select count(*) into s_count from TRIP where TRIP_ID = a_trip_id;
    if s_count = 0 then RAISE_APPLICATION_ERROR(-20002, 'Wycieczka o podanym ID
nie istnieje.');
    end if;

    -- Sprawdzenie czy wycieczka się już nie odbyła
  select TRIP_DATE into s_trip_date from TRIP t where t.TRIP_ID = a_trip_id;
    DBMS_OUTPUT.PUT_LINE('Data wycieczki: ' || s_trip_date);
    if s_trip_date < SYSDATE then RAISE_APPLICATION_ERROR(-20003, 'Wycieczka już
się odbyła.');
    end if;

    -- Sprawdzenie czy osoba jest już zapisana na tą wycieczkę
  select COUNT(*) into s_count from RESERVATION
    where TRIP_ID = a_trip_id and PERSON_ID = a_person_id;
    if s_count > 0 then RAISE_APPLICATION_ERROR(-20004, 'Ta osoba jest już
zapisana na tę wycieczkę.');
    end if;

  select NO_AVAILABLE_PLACES
  into s_available_places
  from TRIP
  where TRIP_ID = a_trip_id;

  if (s_available_places <= 0) then
        RAISE_APPLICATION_ERROR(-20005, 'Brak wolnych miejsc na wycieczce.');
  end if;

    -- Wstawienie rekordu do RESERVATION i pobranie RESERVATION_ID
  insert into RESERVATION (TRIP_ID, PERSON_ID, STATUS)
    values (a_trip_id, a_person_id, 'N')
    returning RESERVATION_ID into s_reservation_id;

    DBMS_OUTPUT.PUT_LINE('Rezerwacja została pomyślnie dodana.');

  P_UPDATE_NO_AVAILABLE_PLACES(a_trip_id);

exception
 when others then rollback;
        DBMS_OUTPUT.PUT_LINE('Wystąpił błąd: ' || SQLERRM);
end;
```

**Modyfikacja procedury p_modify_reservation_status:**

```
create or replace procedure p_modify_reservation_status_6a(a_reservation_id int,
a_status char)
as
```

```
  s_trip_id int;
  s_curr_status char;
  s_available_places int;
begin
  -- Poprawność argumentu a_status
  if a_status not in ('N', 'P', 'C') then
  RAISE_APPLICATION_ERROR(-20001, 'Niepoprawny status.');
    end if;


    -- Pobranie TRIP_ID i STATUS
  select TRIP_ID, STATUS into s_trip_id, s_curr_status
    from RESERVATION where RESERVATION_ID = a_reservation_id;

  select NO_AVAILABLE_PLACES into s_available_places
  from TRIP where TRIP_ID = s_trip_id;



    -- Sprawdzenie istnienia rezerwacji
  if s_trip_id is null then
  RAISE_APPLICATION_ERROR(-20002, 'Rezerwacja o podanym ID nie istnieje.');
    end if;

    -- Sprawdzenie czy rezerwacja nie ma już podanego statusu
  if s_curr_status = a_status then
  RAISE_APPLICATION_ERROR(-20003, 'Rezerwacja ma już ten status.');
    end if;

  if a_status = 'C' then -- anulowanie rezerwacji
    update RESERVATION set STATUS = a_status where RESERVATION_ID =
a_reservation_id;
    DBMS_OUTPUT.PUT_LINE('Twoja rezerwacja została anulowana.');
    P_UPDATE_NO_AVAILABLE_PLACES(s_trip_id);
    return;
  end if;

  if s_curr_status = 'C' then
    if s_available_places <= 0 then
        RAISE_APPLICATION_ERROR(-20004, 'Brak wolnych miejsc');
    end if;

    update RESERVATION set STATUS = a_status where RESERVATION_ID =
a_reservation_id;
    DBMS_OUTPUT.PUT_LINE('Status twojej rezerwacji został zaaktualizowany.');
    P_UPDATE_NO_AVAILABLE_PLACES(s_trip_id);
    return;
  end if;



  if a_status = 'P' then
    update RESERVATION set STATUS = a_status where RESERVATION_ID =
a_reservation_id;
    DBMS_OUTPUT.PUT_LINE('Twoje miejsce jest już potwierdzone i zapłacone.');
```

```
    P_UPDATE_NO_AVAILABLE_PLACES(s_trip_id);
  end if;
end;
```

**Modyfikacja procedury p_modify_max_no_places:**

```
create or replace procedure p_modify_max_no_places_6a(a_trip_id int,
a_max_no_places int)
as
  s_count int;
  s_available_places int;
  s_max_no_places int;
begin
  if a_max_no_places <= 0 then
  RAISE_APPLICATION_ERROR(-20001, 'Proszę podać poprawną liczbę');
    end if;

    select count(*) into s_count from TRIP WHERE TRIP_ID = a_trip_id;
    if s_count = 0 then
  RAISE_APPLICATION_ERROR(-20002, 'Nie istnieje wycieczka o podanym ID');
    end if;

  select NO_AVAILABLE_PLACES
  into s_available_places
  from TRIP
  where TRIP_ID = a_trip_id;

  select MAX_NO_PLACES
  into s_max_no_places
  from TRIP
  where TRIP_ID = a_trip_id;

    if s_max_no_places - s_available_places > a_max_no_places then
  RAISE_APPLICATION_ERROR(-20003, 'Nie można zmiejszyć liczby miejsc poniżej
liczby zarezerwowanych');
    end if;

    UPDATE TRIP
    SET MAX_NO_PLACES = a_max_no_places,
        NO_AVAILABLE_PLACES = s_available_places - s_max_no_places +
a_max_no_places
    WHERE TRIP_ID = a_trip_id;

    P_UPDATE_NO_AVAILABLE_PLACES(a_trip_id);
exception
 when others then rollback;
        raise;
end;
```

## Zadanie 6b.

Triggery do aktualizacji pola `no_available_places`:

**Przy dodawaniu oraz zmianie statusu rezerwacji:**

```
create trigger TRG_UPDATE_TRIP_AVAILABILITY_AFTER_RESERVATION_UPDATE
after update or insert on RESERVATION
for each row
begin
    P_UPDATE_NO_AVAILABLE_PLACES(NEW.TRIP_ID);
end;
```

Gdy procedura wywoła aktualizację rekordu w tabeli RESERVATION, wywołają się powyższe triggery. Jeśli triggery nie wyrzucą błędu to rekord doda się / zaaktualizuję. Po tej akcji wywoła się powyższy trigger (after update) i zaaktualizuje pole no_avaible_places korzystając z procedury P_UPDATE_NO_AVAILABLE_PLACES.

# Zadanie 7.

## Porównaj sposób programowania w systemie Oracle PL/SQL ze znanym ci systemem MS Sqlserver

Języki pomimo różnic są do siebie całkiem podobno jednakże wiele je dzieli:

- Na pewno przydaje się w oracle sql funkcja 'or' np. w zastosowaniach create or replace.
- W ms sql raczej nie było czegoś takiego jak 'for each row' w triggerach.
- Ograniczone jest w oracle'u przypisywanie wartości podzapytania. Trzeba korzystać z formuły 'SELECT ... INTO ...' co jest trochę mniej intuicyjne.
- W funkcja w ms sql jeśli chcieliśmy zwrócić tabelę to mogliśmy po prostu napisać 'return table'. Natomiast tutaj trzeba używać takich form jak 'return SYS_REFCURSOR' co wprowadza dodatkową trudność w przenoszeniu się do oracle sql z innych języków sql.