Oracle PL/SQL,- ćwiczenie
PL/SQL – programowanie proceduralne, widoki, procedury, triggery

Sprawozdanie - Dawid Majchrowski

Rok III Informatyka IEIT

1. Tabele

Zgodnie z poleceniem tworzymy 3 tabele oraz ograniczenia.

- *Tabela wycieczki*

```
CREATE TABLE WYCIECZKI
(
ID_WYCIECZKI INT GENERATED ALWAYS AS IDENTITY NOT NULL
, NAZWA VARCHAR2(100)
, KRAJ VARCHAR2(50)
, DATA DATE
, OPIS VARCHAR2(200)
, LICZBA_MIEJSC INT
, CONSTRAINT WYCIECZKI_PK PRIMARY KEY
(
ID_WYCIECZKI
)
ENABLE
);
```

- *Tabela osoby*

```
CREATE TABLE OSOBY
(
ID_OSOBY INT GENERATED ALWAYS AS IDENTITY NOT NULL
, IMIE VARCHAR2(50)
, NAZWISKO VARCHAR2(50)
, PESEL VARCHAR2(11)
, KONTAKT VARCHAR2(100)
, CONSTRAINT OSOBY_PK PRIMARY KEY
(
ID_OSOBY
)
ENABLE
);
```

- *Tabela Rezerwacje*

```
CREATE TABLE REZERWACJE
(
NR_REZERWACJI INT GENERATED ALWAYS AS IDENTITY NOT NULL
, ID_WYCIECZKI INT
, ID_OSOBY INT
, STATUS CHAR(1)
, CONSTRAINT REZERWACJE_PK PRIMARY KEY
(
NR_REZERWACJI
)
ENABLE
)
```

1a. Ograniczenia

- *Tabela Rezerwacje*

```
ALTER TABLE REZERWACJE
ADD CONSTRAINT REZERWACJE_FK1 FOREIGN KEY
(
ID_OSOBY
)
REFERENCES OSOBY
(
ID_OSOBY
)
ENABLE;

ALTER TABLE REZERWACJE
ADD CONSTRAINT REZERWACJE_FK2 FOREIGN KEY
(
ID_WYCIECZKI
)
REFERENCES WYCIECZKI
(
ID_WYCIECZKI
)
ENABLE;

ALTER TABLE REZERWACJE
ADD CONSTRAINT REZERWACJE_CHK1 CHECK
(status IN ('N','P','Z','A'))
ENABLE;
```

2. Wypełnienie danymi

Zgodnie z poleceniem wypełniamy 3 powyższe tabele danymi:
10 osób, 4 wycieczki, 10 rezerwacji

- *Osoby*

```
INSERT INTO osoby (imie, nazwisko, pesel, kontakt)
VALUES('Adam', 'Kowalski', '87654321', 'tel: 6623');

INSERT INTO osoby (imie, nazwisko, pesel, kontakt)
VALUES('Jan', 'Nowak', '12345678', 'tel: 2312');

INSERT INTO osoby (imie, nazwisko, pesel, kontakt)
VALUES('Jan', 'Towas', '12345679', 'tel: 5123');

INSERT INTO osoby (imie, nazwisko, pesel, kontakt)
VALUES('Damian', 'Pokes', '92343678', 'tel: 1212');

INSERT INTO osoby (imie, nazwisko, pesel, kontakt)
VALUES('Artur', 'Boruc', '72343678', 'tel: 2345');

INSERT INTO osoby (imie, nazwisko, pesel, kontakt)
VALUES('Robert', 'Lewandowski', '72343671', 'tel: 1331');

INSERT INTO osoby (imie, nazwisko, pesel, kontakt)
VALUES('Marcin', 'Gortat', '12121212', 'tel: 1111');

INSERT INTO osoby (imie, nazwisko, pesel, kontakt)
VALUES('Robert', 'Kubica', '42143121', 'tel: 3333');

INSERT INTO osoby (imie, nazwisko, pesel, kontakt)
VALUES('Jan', 'Kowalski', '72343671', 'tel: 1666');

INSERT INTO osoby (imie, nazwisko, pesel, kontakt)
VALUES('Janusz', 'Borek', '12312314', 'tel: 5331');
```

- *Wycieczki*

```
INSERT INTO wycieczki (nazwa, kraj, data, opis, liczba_miejsc)
VALUES ('Wycieczka do Paryza','Francja',TO_DATE('2016-01-01','YYYY-MM-
DD'),'Ciekawa wycieczka ...',3);

INSERT INTO wycieczki (nazwa, kraj, data, opis, liczba_miejsc)
VALUES ('Piękny Kraków','Polska',TO_DATE('2017-02-03','YYYY-MM-DD'),'Najciekawa
wycieczka ...',2);

INSERT INTO wycieczki (nazwa, kraj, data, opis, liczba_miejsc)
VALUES ('Wieliczka','Polska',TO_DATE('2017-03-03','YYYY-MM-DD'),'Zadziwiająca
kopalnia ...',2);

INSERT INTO wycieczki (nazwa, kraj, data, opis, liczba_miejsc)
VALUES ('Energylandia','Polska',TO_DATE('2019-10-28','YYYY-MM-DD'),'Zadziwiający
park rozrywki ...',12);
```

- *Rezerwacje*

```
INSERT INTO rezerwacje(id_wycieczki, id_osoby, status)
VALUES (1,1,'N');

INSERT INTO rezerwacje(id_wycieczki, id_osoby, status)
VALUES (2,2,'P');

INSERT INTO rezerwacje(id_wycieczki, id_osoby, status)
VALUES (3,3,'Z');

INSERT INTO rezerwacje(id_wycieczki, id_osoby, status)
VALUES (4,4,'Z');

INSERT INTO rezerwacje(id_wycieczki, id_osoby, status)
VALUES (4,5,'A');

INSERT INTO rezerwacje(id_wycieczki, id_osoby, status)
VALUES (3,6,'P');

INSERT INTO rezerwacje(id_wycieczki, id_osoby, status)
VALUES (4,7,'N');

INSERT INTO rezerwacje(id_wycieczki, id_osoby, status)
VALUES (1,8,'A');

INSERT INTO rezerwacje(id_wycieczki, id_osoby, status)
VALUES (2,9,'N');

INSERT INTO rezerwacje(id_wycieczki, id_osoby, status)
VALUES (4,10,'Z');
```

3. Widoki

Zgodnie z poleceniem tworzymy 6 kolejnych widoków. (od tej pory wszystkie tworzone obiekty będą się zaczynać od „CREATE OR REPLACE”, dzięki czemu nie musimy usuwać obiektów w razie pomyłki przy tworzeniu oraz zaoszczędzi czas.)
Dane widoki rozbudowujemy od ID, gdyż będziemy korzystać z widoków w dalszej części.

- *Wycieczki osoby*

```
CREATE OR REPLACE VIEW WYCIECZKI_OSOBY
AS
SELECT
o.ID_OSOBY,
w.ID_WYCIECZKI,
w.NAZWA,
w.KRAJ,
w.DATA,
o.IMIE,
o.NAZWISKO,
r.STATUS
FROM WYCIECZKI w
JOIN REZERWACJE r ON w.ID_WYCIECZKI = r.ID_WYCIECZKI
JOIN OSOBY o ON r.ID_OSOBY = o.ID_OSOBY;
```

- *Wycieczki osoby potwierdzone*

Zakładamy, że interesują nas również przeszłe wycieczki

```
CREATE OR REPLACE VIEW WYCIECZKI_OSOBY_POTWIERDZONE
AS
SELECT
w.ID_WYCIECZKI,
w.NAZWA,
w.KRAJ,
w.DATA,
o.IMIE,
o.NAZWISKO,
r.STATUS
FROM WYCIECZKI w
JOIN REZERWACJE r ON w.ID_WYCIECZKI = r.ID_WYCIECZKI
JOIN OSOBY o ON r.ID_OSOBY = o.ID_OSOBY
WHERE r.STATUS IN ('P', 'Z');
```

- *Wycieczki przyszłe (Przyszłe wycieczki osób, które nie anulowały rezerwacji)*

```
CREATE OR REPLACE VIEW WYCIECZKI_PRZYSZLE
AS
SELECT
w.ID_WYCIECZKI,
w.NAZWA,
w.KRAJ,
w.DATA,
o.IMIE,
o.NAZWISKO,
r.STATUS
FROM WYCIECZKI w
JOIN REZERWACJE r ON w.ID_WYCIECZKI = r.ID_WYCIECZKI
JOIN OSOBY o ON r.ID_OSOBY = o.ID_OSOBY
WHERE w.DATA > CURRENT_DATE AND r.STATUS <> 'A';
```

- *Wycieczki miejsca*

Wycieczki z aktualną liczbą wolnych miejsc. Zakładamy, że wprowadzone dane są poprawne i liczba wolnych miejsc musi być nieujemna(Kontrola w procedurach/triggerach).

```
CREATE OR REPLACE VIEW WYCIECZKI_MIEJSCA
AS
SELECT
w.ID_WYCIECZKI,
w.KRAJ,
w.DATA,
w.NAZWA,
w.OPIS,
w.LICZBA_MIEJSC,
w.LICZBA_MIEJSC - (SELECT COUNT(*) FROM REZERWACJE r WHERE
w.ID_WYCIECZKI = r.ID_WYCIECZKI AND r.STATUS <> 'A') AS
LICZBA_WOLNYCH_MIEJSC
FROM WYCIECZKI w;
```

- *Dostępne Wycieczki (Wycieczki z dodatnią liczbą wolnych miejsc i przyszłą datą)*

```
CREATE OR REPLACE VIEW DOSTEPNE_WYCIECZKI
AS
SELECT
w.ID_WYCIECZKI,
w.KRAJ,
w.DATA,
w.NAZWA,
w.OPIS,
w.LICZBA_MIEJSC,
w.LICZBA_WOLNYCH_MIEJSC
FROM WYCIECZKI_MIEJSCA w
WHERE LICZBA_WOLNYCH_MIEJSC > 0 AND w.DATA > CURRENT_DATE;
```

- *Rezerwacje do anulowania*

(lista niepotwierdzonych rezerwacji które powinne zostać
anulowane, rezerwacje przygotowywane są do anulowania na tydzień przed wyjazdem)

```
CREATE OR REPLACE VIEW REZERWACJE_DO_ANULOWANIA
AS
SELECT
r.NR_REZERWACJI,
w.ID_WYCIECZKI,
w.NAZWA,
w.KRAJ,
w.DATA,
r.STATUS
FROM REZERWACJE r
JOIN WYCIECZKI W ON r.ID_WYCIECZKI = W.ID_WYCIECZKI
WHERE  r.STATUS = 'N' AND w.DATA BETWEEN CURRENT_DATE AND
CURRENT_DATE+7;
```

4. Funkcje

4a. Typy danych

Tworzone funkcję będą zwracać tabele jako typy danych, dlatego zanim stworzymy funkcje, stworzymy 2 nowe typy danych.

- *Uczestnicy (typ danych dla pierwszych trzech funkcji)*

```
CREATE OR REPLACE TYPE UCZESTNICY_DATA AS OBJECT (
  ID_OSOBY      INT
, ID_WYCIECZKI  INT
, NAZWA         VARCHAR2(100)
, KRAJ          VARCHAR2(50)
, "DATA"        DATE
, IMIE          VARCHAR2(50)
, NAZWISKO      VARCHAR2(50)
, STATUS        CHAR(1)
);

CREATE OR REPLACE TYPE UCZESTNICY_TABLE IS TABLE OF
UCZESTNICY_DATA;
```

- *Wycieczki (typ danych dla czwartej funkcji)*

```
CREATE OR REPLACE TYPE WYCIECZKI_DATA AS OBJECT (
  ID_WYCIECZKI            INT
, NAZWA                   VARCHAR2(100)
, KRAJ                    VARCHAR2(50)
, "DATA"                  DATE
, OPIS                    VARCHAR2(100)
, LICZBA_MIEJSC           INT
, POZOSTALA_LICZBA_MIEJSC   INT
);

CREATE OR REPLACE TYPE WYCIECZKI_TABLE IS TABLE OF
WYCIECZKI_DATA;
```

4b. Funkcje

- *Uczestnicy wycieczki*

```
CREATE OR REPLACE FUNCTION UCZESTNICY_WYCIECZKI(ID INT)
RETURN UCZESTNICY_TABLE AS
   v_ret UCZESTNICY_TABLE;
   trip_not_found EXCEPTION;
   trip_count INT;
BEGIN
   SELECT COUNT(*) INTO trip_count FROM WYCIECZKI WHERE
WYCIECZKI.ID_WYCIECZKI = ID;
   IF trip_count = 0 THEN
      RAISE trip_not_found;
   END IF;
   SELECT UCZESTNICY_DATA(v.ID_OSOBY, v.ID_WYCIECZKI, v.NAZWA, v.KRAJ,
v.DATA, v.IMIE, v.NAZWISKO, v.STATUS)
 BULK COLLECT INTO v_ret
 FROM WYCIECZKI_OSOBY v WHERE v.ID_WYCIECZKI = ID AND v.STATUS <> 'A';
RETURN v_ret;
END UCZESTNICY_WYCIECZKI;
```

- *Rezerwacje osoby*

```
CREATE OR REPLACE FUNCTION REZERWACJE_OSOBY(ID INT)
RETURN UCZESTNICY_TABLE AS
   v_ret UCZESTNICY_TABLE;
   id_not_found EXCEPTION;
   id_count INT;
BEGIN
   SELECT COUNT(*) INTO id_count FROM OSOBY o WHERE o.ID_OSOBY = ID;
   IF id_count = 0 THEN
      RAISE id_not_found;
   END IF;
   SELECT UCZESTNICY_DATA(v.ID_OSOBY, v.ID_WYCIECZKI, v.NAZWA,
v.KRAJ, v.DATA, v.IMIE, v.NAZWISKO, v.STATUS)
   BULK COLLECT INTO v_ret
   FROM WYCIECZKI_OSOBY v WHERE v.ID_OSOBY = ID;
RETURN v_ret;
END REZERWACJE_OSOBY;
```

- *Przyszłe rezerwacje osoby*

```
CREATE OR REPLACE FUNCTION PRZYSZLE_REZERWACJE_OSOBY(ID INT)
RETURN UCZESTNICY_TABLE AS
   v_ret UCZESTNICY_TABLE;
   id_not_found EXCEPTION;
   id_count INT;
BEGIN
   SELECT COUNT(*) INTO id_count FROM OSOBY o WHERE o.ID_OSOBY = ID;
   IF id_count = 0 THEN
      RAISE id_not_found;
   END IF;
   SELECT UCZESTNICY_DATA(v.ID_OSOBY, v.ID_WYCIECZKI, v.NAZWA,
v.KRAJ, v.DATA, v.IMIE, v.NAZWISKO, v.STATUS)
   BULK COLLECT INTO v_ret
   FROM WYCIECZKI_OSOBY v WHERE v.ID_OSOBY = ID AND v.STATUS <> 'A'
AND v.DATA > CURRENT_DATE;
RETURN v_ret;
END PRZYSZLE_REZERWACJE_OSOBY;
```

- *Dostępne wycieczki*

```
CREATE OR REPLACE FUNCTION DOSTEPNE_WYCIECZKI_KRAJ(KR VARCHAR,
OD DATE, DO DATE) RETURN WYCIECZKI_TABLE AS
   v_ret WYCIECZKI_TABLE;
   country_not_found EXCEPTION;
   wrong_date EXCEPTION;
   country_count INT;
BEGIN
   SELECT COUNT(*) INTO country_count FROM WYCIECZKI w WHERE w.KRAJ =
KR;
   IF country_count = 0 THEN
      RAISE country_not_found;
   END IF;

   IF OD > DO THEN
      RAISE wrong_date;
   END IF;

   IF DO < CURRENT_DATE THEN
      RAISE wrong_date;
   END IF;

   SELECT WYCIECZKI_DATA(v.ID_WYCIECZKI, v.NAZWA, v.KRAJ, v.DATA,
v.OPIS, v.LICZBA_MIEJSC, v.liczba_wolnych_miejsc)
   BULK COLLECT INTO v_ret
   FROM WYCIECZKI_MIEJSCA v WHERE v.LICZBA_WOLNYCH_MIEJSC > 0 AND
v.KRAJ = KR AND v.DATA BETWEEN OD AND DO;
RETURN v_ret;
END DOSTEPNE_WYCIECZKI_KRAJ;
```

5. Procedury

- *Dodaj rezerwacje*

```
CREATE OR REPLACE PROCEDURE DODAJ_REZERWACJE
(ID_WYC INT, ID_OS INT) AS
   counter INT;
   not_exists EXCEPTION;
   already_exists EXCEPTION;
BEGIN
   SAVEPOINT DODAJ_REZERWACJE_SAVEPOINT;
      SELECT COUNT(*) INTO counter FROM OSOBY o WHERE o.ID_OSOBY = ID_OS;
   IF counter = 0 THEN
      RAISE not_exists;
   END IF;

   SELECT COUNT(*) INTO counter FROM dostepne_wycieczki w WHERE
w.ID_WYCIECZKI = ID_WYC;
   IF counter = 0
   THEN
      RAISE not_exists; -- No trip available OR no seats avaliable
   END IF;

   SELECT COUNT(*) INTO counter FROM REZERWACJE r WHERE r.ID_WYCIECZKI
= ID_WYC AND r.ID_OSOBY = ID_OS;
   IF counter > 0
   THEN
      RAISE already_exists;
   END IF;

   INSERT INTO REZERWACJE (id_wycieczki, id_osoby, STATUS)
   VALUES (ID_WYC, ID_OS, 'N');
   COMMIT;

   EXCEPTION
     WHEN OTHERS THEN
     ROLLBACK TO DODAJ_REZERWACJE_SAVEPOINT;
     RAISE;
END DODAJ_REZERWACJE;
```

- *Zmień status rezerwacji*

```
CREATE OR REPLACE PROCEDURE ZMIEN_STATUS_REZERWACJI
(ID_REZ INT, NOWY_STATUS CHAR) AS
   counter INT;
   stary_status CHAR;
   not_exists EXCEPTION;
   wrong_status EXCEPTION;
BEGIN
   SAVEPOINT ZMIEN_STATUS_REZERWACJI_SAVEPOINT;

   SELECT COUNT(*) INTO counter FROM REZERWACJE r WHERE
r.NR_REZERWACJI = ID_REZ;
   IF counter = 0 THEN
      RAISE not_exists;
   END IF;

   SELECT COUNT(*) INTO counter FROM REZERWACJE r
         JOIN WYCIECZKI w ON r.ID_WYCIECZKI = w.ID_WYCIECZKI
       WHERE r.NR_REZERWACJI = ID_REZ AND w.DATA > CURRENT_DATE;
   IF counter = 0 THEN
     RAISE not_exists; -- Only future trips
   END IF;

   SELECT STATUS into stary_status FROM REZERWACJE r WHERE
r.NR_REZERWACJI = ID_REZ;

   IF stary_status='A' THEN
      SELECT DW.LICZBA_WOLNYCH_MIEJSC INTO counter FROM REZERWACJE r
         JOIN DOSTEPNE_WYCIECZKI DW ON r.ID_WYCIECZKI =
DW.ID_WYCIECZKI
       WHERE r.NR_REZERWACJI = ID_REZ;
       IF counter < 1 THEN
         RAISE wrong_status; -- No seats left for this reservation
       END IF;
   END IF;

   UPDATE REZERWACJE R
   SET R.STATUS = NOWY_STATUS
   WHERE R.NR_REZERWACJI = ID_REZ;
   COMMIT;

   EXCEPTION
    WHEN OTHERS THEN
    ROLLBACK TO ZMIEN_STATUS_REZERWACJI_SAVEPOINT;
    RAISE;

END ZMIEN_STATUS_REZERWACJI;
```

- *Zmień liczbę miejsc*

```
CREATE OR REPLACE PROCEDURE ZMIEN_LICZBE_MIEJSC
(ID_WYC INT, NOWA_LICZBA_MIEJSC INT) AS
   counter INT;
   not_exists EXCEPTION;
   wrong_size EXCEPTION;
BEGIN
   SAVEPOINT ZMIEN_LICZBE_MIEJSC_SAVEPOINT;

   SELECT COUNT(*) INTO counter FROM WYCIECZKI w WHERE w.ID_WYCIECZKI
= ID_WYC AND w.DATA > CURRENT_DATE;

   IF counter = 0 THEN
      RAISE not_exists; -- only future trips
   END IF;

   SELECT w.LICZBA_MIEJSC - w.LICZBA_WOLNYCH_MIEJSC INTO counter FROM
DOSTEPNE_WYCIECZKI w WHERE w.ID_WYCIECZKI = ID_WYC; -- number of taken
seats

   IF counter > NOWA_LICZBA_MIEJSC THEN -- number of taken seats should be <= new
number of seats
      RAISE wrong_size; -- Not enough seats for existing reservations.
   END IF;

   UPDATE WYCIECZKI w
   SET w.LICZBA_MIEJSC = NOWA_LICZBA_MIEJSC
   WHERE w.ID_WYCIECZKI = ID_WYC;
   COMMIT;

   EXCEPTION
    WHEN OTHERS THEN
    ROLLBACK TO ZMIEN_LICZBE_MIEJSC_SAVEPOINT;
    RAISE;

END ZMIEN_LICZBE_MIEJSC;
```

6. Tabela dziennikującą

- *Tabela*

```
CREATE TABLE REZERWACJE_LOG
(
ID INT GENERATED ALWAYS AS IDENTITY NOT NULL
, ID_REZERWACJI INT
, DATA DATE
, STATUS CHAR(1)
, CONSTRAINT REZERWACJE_LOG_PK PRIMARY KEY
(
ID
)
ENABLE
);

ALTER TABLE REZERWACJE_LOG
ADD CONSTRAINT REZERWACJE_LOG_FK FOREIGN KEY
(
ID_REZERWACJI
)
REFERENCES REZERWACJE
(
NR_REZERWACJI
)
ENABLE;
```

- *Zmiany*

W procedurze „dodaj rezerwacje", przed słowem „COMMIT" dodajemy:

```
...
   SELECT r.NR_REZERWACJI INTO counter FROM REZERWACJE r WHERE
r.ID_WYCIECZKI = ID_WYC AND r.ID_OSOBY=ID_OS;
   INSERT INTO REZERWACJE_LOG (id_rezerwacji, data, status)
   VALUES (counter, CURRENT_DATE, 'N');
   COMMIT;
...
```

W procedurze „zmień status rezerwacji", przed słowem „COMMIT" dodajemy:

```
 ...
   INSERT INTO REZERWACJE_LOG (id_rezerwacji, data, status)
  VALUES (ID_REZ, CURRENT_DATE, NOWY_STATUS);
  COMMIT;
 ...
```

7. Zmiana struktury bazy danych

- *Nowa kolumna w tabeli wycieczki*

```
ALTER TABLE WYCIECZKI
ADD LICZBA_WOLNYCH_MIEJSC INT;
```

- *Dodajemy c*onstraint*, który gwarantuje niezerową liczbę wolnych miejsc przy wstawianiu.*

```
ALTER TABLE WYCIECZKI
ADD CONSTRAINT NIEZEROWA_LICZBA_WOLNYCH_MIEJSC CHECK
  (LICZBA_WOLNYCH_MIEJSC > 0);
```

7a) Widoki

- *Wycieczki Miejsca*

```
CREATE OR REPLACE VIEW WYCIECZKI_MIEJSCA_2
AS
SELECT
w.ID_WYCIECZKI,
w.KRAJ,
w.DATA,
w.NAZWA,
w.OPIS,
w.LICZBA_MIEJSC,
w.LICZBA_WOLNYCH_MIEJSC
FROM WYCIECZKI w;
```

- *Dostępne Wycieczki*

```
CREATE OR REPLACE VIEW DOSTEPNE_WYCIECZKI_2
AS
SELECT
w.ID_WYCIECZKI,
w.KRAJ,
w.DATA,
w.NAZWA,
w.OPIS,
w.LICZBA_MIEJSC,
w.LICZBA_WOLNYCH_MIEJSC
FROM WYCIECZKI w
WHERE LICZBA_WOLNYCH_MIEJSC > 0 AND w.DATA > CURRENT_DATE;
```

7b) Procedury

Zgodnie z poleceniem dodajemy procedurę przelicz oraz edytujemy 3 istniejące procedury, jeżeli posiadały one logowanie w tabeli dziennikującej to ją zostawiamy.

- *Przelicz*

(Update bez klauzuli WHERE zaktualizuje wartości w całej tabeli po jednym użyciu)

```
CREATE OR REPLACE PROCEDURE PRZELICZ AS
BEGIN
   SAVEPOINT PRZELICZ_SAVEPOINT;

   UPDATE WYCIECZKI w
   SET w.LICZBA_WOLNYCH_MIEJSC = w.LICZBA_MIEJSC - (SELECT COUNT(*)
FROM REZERWACJE r WHERE r.ID_WYCIECZKI = w.ID_WYCIECZKI AND
r.STATUS <> 'A');
   COMMIT;

   EXCEPTION
     WHEN OTHERS THEN
     ROLLBACK TO PRZELICZ_SAVEPOINT;
     RAISE;
END PRZELICZ;
```

- *Dodaj rezerwacje*

```
CREATE OR REPLACE PROCEDURE DODAJ_REZERWACJE_2
(ID_WYC INT, ID_OS INT) AS
   counter INT;
   not_exists EXCEPTION;
   already_exists EXCEPTION;
BEGIN
   SAVEPOINT DODAJ_REZERWACJE_2_SAVEPOINT;
   SELECT COUNT(*) INTO counter FROM OSOBY WHERE OSOBY.ID_OSOBY =
ID_OS;
   IF counter = 0 THEN
      RAISE not_exists;
   END IF;

   SELECT COUNT(*) INTO counter FROM WYCIECZKI w WHERE w.ID_WYCIECZKI
= ID_WYC AND w.LICZBA_WOLNYCH_MIEJSC > 0
AND w.DATA > CURRENT_DATE;
   IF counter = 0 THEN
      RAISE not_exists; -- No seats for given trip
   END IF;

   SELECT COUNT(*) INTO counter FROM REZERWACJE r WHERE r.ID_WYCIECZKI
= ID_WYC AND r.ID_OSOBY = ID_OS;
   IF counter > 0 THEN
      RAISE already_exists;
   END IF;

   INSERT INTO REZERWACJE (id_wycieczki, id_osoby, STATUS)
   VALUES (ID_WYC, ID_OS, 'N');

   SELECT r.NR_REZERWACJI INTO counter FROM REZERWACJE r WHERE
r.ID_WYCIECZKI = ID_WYC AND r.ID_OSOBY=ID_OS;
   INSERT INTO REZERWACJE_LOG (id_rezerwacji, data, status)
   VALUES (counter, CURRENT_DATE, 'N');
   COMMIT;

   EXCEPTION
    WHEN OTHERS THEN
    ROLLBACK TO DODAJ_REZERWACJE_2_SAVEPOINT;
    RAISE;
END DODAJ_REZERWACJE_2;
```

- *Zmień status rezerwacji*

```
CREATE OR REPLACE PROCEDURE ZMIEN_STATUS_REZERWACJI_2(ID_REZ INT,
NOWY_STATUS CHAR) AS
   counter INT;
   stary_status CHAR;
   not_exists EXCEPTION;
   wrong_status EXCEPTION;
BEGIN
   SAVEPOINT ZMIEN_STATUS_REZERWACJI_2_SAVEPOINT;

   SELECT COUNT(*) INTO counter FROM REZERWACJE r WHERE
r.NR_REZERWACJI = ID_REZ;
   IF counter = 0 THEN
      RAISE not_exists;
   END IF;

   SELECT COUNT(*) INTO counter FROM REZERWACJE r
         JOIN WYCIECZKI w ON r.ID_WYCIECZKI = w.ID_WYCIECZKI
      WHERE r.NR_REZERWACJI = ID_REZ AND w.DATA > CURRENT_DATE;
   IF counter = 0 THEN
     RAISE not_exists; -- Only future trips
   END IF;

   SELECT STATUS into stary_status FROM REZERWACJE r WHERE
r.NR_REZERWACJI = ID_REZ;

   IF stary_status='A' THEN
     SELECT W.LICZBA_WOLNYCH_MIEJSC INTO counter FROM REZERWACJE r
        JOIN WYCIECZKI W ON r.ID_WYCIECZKI = W.ID_WYCIECZKI
      WHERE r.NR_REZERWACJI = ID_REZ;
      IF counter < 1 THEN
        RAISE wrong_status; -- No seats left for this reservation
      END IF;
   END IF;

   UPDATE REZERWACJE R
   SET R.STATUS = NOWY_STATUS
   WHERE R.NR_REZERWACJI = ID_REZ;

   INSERT INTO REZERWACJE_LOG (id_rezerwacji, data, status)
   VALUES (ID_REZ, CURRENT_DATE, NOWY_STATUS);
   COMMIT;

   EXCEPTION
    WHEN OTHERS THEN
    ROLLBACK TO ZMIEN_STATUS_REZERWACJI_2_SAVEPOINT;
    RAISE;

END ZMIEN_STATUS_REZERWACJI_2;
```

- *Zmień liczbę miejsc*

```
CREATE OR REPLACE PROCEDURE ZMIEN_LICZBE_MIEJSC_2
(ID_WYC INT, NOWA_LICZBA_MIEJSC INT) AS
   counter INT;
   not_exists EXCEPTION;
   wrong_size EXCEPTION;
BEGIN
   SAVEPOINT ZMIEN_LICZBE_MIEJSC_2_SAVEPOINT;

   SELECT COUNT(*) INTO counter FROM WYCIECZKI w WHERE w.ID_WYCIECZKI
= ID_WYC AND w.DATA > CURRENT_DATE;

   IF counter = 0 THEN
      RAISE not_exists; -- only future trips
   END IF;

   SELECT w.LICZBA_MIEJSC - w.LICZBA_WOLNYCH_MIEJSC INTO counter FROM
WYCIECZKI w WHERE w.ID_WYCIECZKI = ID_WYC; -- number of taken seats

   IF counter > NOWA_LICZBA_MIEJSC THEN -- number of taken seats should be <= new
number of seats
      RAISE wrong_size; -- Not enough seats for existing reservations.
   END IF;

   UPDATE WYCIECZKI w
   SET
      w.LICZBA_MIEJSC = NOWA_LICZBA_MIEJSC,
      w.LICZBA_WOLNYCH_MIEJSC = NOWA_LICZBA_MIEJSC - counter
   WHERE w.ID_WYCIECZKI = ID_WYC;
   COMMIT;

   EXCEPTION
    WHEN OTHERS THEN
    ROLLBACK TO ZMIEN_LICZBE_MIEJSC_2_SAVEPOINT;
    RAISE;

END ZMIEN_LICZBE_MIEJSC_2;
```

8. Zmiana strategii zapisywania do dziennika rezerwacji

8a) Triggery

- *Dodanie rezerwacji*

```
CREATE OR REPLACE TRIGGER NOWA_REZERWACJA_TRIGGER
  AFTER INSERT
  ON REZERWACJE
  FOR EACH ROW
  BEGIN
    INSERT INTO REZERWACJE_LOG (ID_REZERWACJI, DATA, STATUS)
    VALUES (:NEW.NR_REZERWACJI, CURRENT_DATE, :NEW.STATUS);
END NOWA_REZERWACJA_TRIGGER;
```

- *Zmiana statusu*

```
CREATE OR REPLACE TRIGGER ZMIANA_STATUSU_TRIGGER
  AFTER UPDATE
  ON REZERWACJE
  FOR EACH ROW
  BEGIN
    INSERT INTO REZERWACJE_LOG (ID_REZERWACJI, DATA, STATUS)
    VALUES (:NEW.NR_REZERWACJI, CURRENT_DATE, :NEW.STATUS);
  END ZMIANA_STATUSU_TRIGGER;
```

- *Zabronienie usunięcia rezerwacji*

```
CREATE OR REPLACE TRIGGER
ZABRONIENIE_USUNIECIA_REZERWACJI_TRIGGER
  BEFORE DELETE
  ON REZERWACJE
  FOR EACH ROW
  DECLARE
     not_allowed EXCEPTION;
  BEGIN
    RAISE not_allowed;
  END ZABRONIENIE_USUNIECIA_REZERWACJI_TRIGGER;
```

8b) Procedury (Zmiany)
Usuwamy dodane w **poleceniu 6** następujące linie (wszystko przed słowem COMMIT) z następujących procedur.

- *Dodaj rezerwacje (następujące linijki usuwany, z wyłączeniem COMMIT)*

```
...
   SELECT r.NR_REZERWACJI INTO counter FROM REZERWACJE r WHERE
r.ID_WYCIECZKI = ID_WYC AND r.ID_OSOBY=ID_OS;
   INSERT INTO REZERWACJE_LOG (id_rezerwacji, data, status)
   VALUES (counter, CURRENT_DATE, 'N');
   COMMIT;
...
```

- *Zmień status rezerwacji (następujące linijki usuwany, z wyłączeniem COMMIT)*

```
...
  INSERT INTO REZERWACJE_LOG (id_rezerwacji, data, status)
 VALUES (ID_REZ, CURRENT_DATE, NOWY_STATUS);
 COMMIT;
...
```

9. Zmiana strategii obsługi pola liczba wolnych miejsc

9a) Triggery

- *Dodanie rezerwacji (Nie martwimy się, czy liczba wolnych miejsc spadnie poniżej 0, ponieważ w tabeli wycieczki dodaliśmy constraint sprawdzający ten warunek)*

```
CREATE OR REPLACE TRIGGER NOWA_REZERWACJA_TRIGGER
 AFTER INSERT
 ON REZERWACJE
 FOR EACH ROW
 BEGIN
  INSERT INTO REZERWACJE_LOG (ID_REZERWACJI, DATA, STATUS)
  VALUES (:NEW.NR_REZERWACJI, CURRENT_DATE, :NEW.STATUS);

  UPDATE WYCIECZKI w
  SET LICZBA_WOLNYCH_MIEJSC = LICZBA_WOLNYCH_MIEJSC - 1
  WHERE w.ID_WYCIECZKI = :NEW.ID_WYCIECZKI;
END NOWA_REZERWACJA_TRIGGER;
```

- *Zmiana statusu*

```
CREATE OR REPLACE TRIGGER ZMIANA_STATUSU_TRIGGER
  AFTER UPDATE
  ON REZERWACJE
  FOR EACH ROW
  DECLARE
    to_add INT;
  BEGIN
    INSERT INTO REZERWACJE_LOG (ID_REZERWACJI, DATA, STATUS)
    VALUES (:NEW.NR_REZERWACJI, CURRENT_DATE, :NEW.STATUS);

    CASE
      WHEN :OLD.STATUS <> 'A' AND :NEW.STATUS = 'A'
      THEN
        to_add := 1;
      WHEN :OLD.STATUS = 'A' AND :NEW.STATUS <> 'A'
      THEN
        to_add := -1;
    ELSE
      to_add := 0;
    END CASE;

    UPDATE WYCIECZKI w
    SET LICZBA_WOLNYCH_MIEJSC = LICZBA_WOLNYCH_MIEJSC + to_add
    WHERE w.ID_WYCIECZKI = :NEW.ID_WYCIECZKI;
  END ZMIANA_STATUSU_TRIGGER;
```

- *Zmiana liczby miejsc na poziomie wycieczki*

```
CREATE OR REPLACE TRIGGER ZMIANA_MIEJSC_TRIGGER
  BEFORE UPDATE OF LICZBA_MIEJSC
  ON WYCIECZKI
  FOR EACH ROW
  BEGIN
    SELECT :OLD.LICZBA_WOLNYCH_MIEJSC +
          (:NEW.LICZBA_MIEJSC - :OLD.LICZBA_MIEJSC) INTO
:NEW.LICZBA_WOLNYCH_MIEJSC
    FROM DUAL;
  END ZMIANA_MIEJSC_TRIGGER;
```

9b) Procedury

- *Dodaj rezerwacje*
- *Zmień status rezerwacji*
- *Zmień liczbę miejsc*