Sprawozdanie 1 Oracle PL/SQL

data laboratorium: 16.10.19 r.
data odddania: 30.10.19 r.

Mateusz Nabywaniec

1. **Widoki**

a) wycieczki_osoby

```
CREATE VIEW wycieczki_osoby
   AS
       SELECT
           w.ID_WYCIECZKI,
           w.NAZWA,
           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;
```

b) Wycieczki_osoby_potwierdzone

```
CREATE VIEW wycieczki_osoby_potwierdzone
   AS SELECT
       w.ID_WYCIECZKI,
       w.KRAJ,
       w.DATA,
       w.NAZWA,
       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 = 'P';
```

c) Wycieczki_przyszłe

```
CREATE VIEW wycieczki_przyszle
   AS select
```

```
      w.ID_WYCIECZKI,
      w.KRAJ,
      w.DATA,
      w.NAZWA,
      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;
```

d) Wycieczki_miejsca

```
CREATE VIEW WYCIECZKI_MIEJSCA
AS
  SELECT
      w.ID_WYCIECZKI,
      w.KRAJ,
      w.NAZWA,
      w.LICZBA_MIEJSC,
      w.liczba_miejsc-
      COUNT(*) as LICZBA_WOLNYCH_MIEJSC
  FROM WYCIECZKI w
      LEFT JOIN REZERWACJE r ON r.ID_WYCIECZKI = w.ID_WYCIECZKI
       WHERE r.STATUS != 'A'
  GROUP BY w.ID_WYCIECZKI, w.KRAJ, w.NAZWA, w.LICZBA_MIEJSC;
```

e) Dostępne_wycieczki

```
CREATE VIEW DOSTEPNE_WYCIECZKI_VIEW
AS
  SELECT
      wm.ID_WYCIECZKI,
      wm.KRAJ,
      wm.NAZWA,
      wm.LICZBA_MIEJSC,
      wm.LICZBA_WOLNYCH_MIEJSC
  FROM WYCIECZKI_MIEJSCA wm
  JOIN WYCIECZKI W on wm.ID_WYCIECZKI = W.ID_WYCIECZKI
      WHERE wm.LICZBA_WOLNYCH_MIEJSC > 0 AND CURRENT_DATE > w.DATA;
```

f) rezerwacje_do_anulowania

```
CREATE VIEW REZERWACJE_DO_ANULOWANIA
AS
   SELECT
       r.NR_REZERWACJI,
       w.NAZWA,
       w.DATA,
       r.STATUS,
       o.IMIE,
       o.NAZWISKO
   FROM REZERWACJE r
       JOIN WYCIECZKI w on r.ID_WYCIECZKI = w.ID_WYCIECZKI
       JOIN OSOBY o on r.ID_OSOBY = o.ID_OSOBY
       WHERE (CURRENT_DATE - 7) > w.DATA
           AND r.STATUS = 'N';
```

2. Procedury / funkcje pobierające dane

a) Uczestnicy wycieczki

Tworzę najpierw nowy typ - wiersz tabeli uczestnicy wycieczki

```
create TYPE uczestnicy_wycieczki_row AS OBJECT (
ID_WYCIECZKI INT,
NAZWA VARCHAR2(100),
KRAJ VARCHAR2(50),
DATA DATE,
IMIE VARCHAR2(50),
NAZWISKO VARCHAR2(50),
STATUS CHAR(1)
)
/
```

```
create type UCZESTNICY_WYCIECZKI_TABLE as table of UCZESTNICY_WYCIECZKI_ROW
/
```

```
CREATE OR REPLACE
FUNCTION uczestnicy_wycieczki(id_wyc INT)
```

```
 return UCZESTNICY_WYCIECZKI_TABLE as v_ret UCZESTNICY_WYCIECZKI_TABLE;
 istnieje                          integer;
 BEGIN
   SELECT COUNT(*)  INTO istnieje FROM WYCIECZKI
   WHERE WYCIECZKI.ID_WYCIECZKI = id_wyc;

   IF istnieje = 0 THEN
     raise_application_error(-20001, 'Brak wycieczki o podanym id');
   END IF;

   SELECT uczestnicy_wycieczki_row(w.ID_WYCIECZKI,  w.NAZWA, w.KRAJ,
w.DATA, o.IMIE,
                           o.NAZWISKO, r.STATUS)
       BULK COLLECT INTO v_ret
   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.ID_WYCIECZKI = id_wyc;
   return v_ret;
 end uczestnicy_wycieczki2;
```

b) rezerwacje_osoby

```
create FUNCTION rezerwacje_osoby(id_os INT)
   RETURN UCZESTNICY_WYCIECZKI_TABLE as v_ret UCZESTNICY_WYCIECZKI_TABLE;
       os_istnieje                              INTEGER;
   BEGIN
       SELECT COUNT(*) INTO os_istnieje FROM OSOBY
       WHERE OSOBY.ID_OSOBY = id_os;

       IF os_istnieje = 0 THEN
         raise_application_error(-20001, 'Brak osoby o podanym id');
       END IF;

       SELECT uczestnicy_wycieczki_row(w.ID_WYCIECZKI,  w.NAZWA, w.KRAJ,
w.DATA, o.IMIE,
                           o.NAZWISKO, r.STATUS)
       BULK COLLECT INTO v_ret
       FROM WYCIECZKI w
          JOIN REZERWACJE r ON w.ID_WYCIECZKI = r.ID_WYCIECZKI
```

```
          JOIN OSOBY o ON r.ID_OSOBY = o.ID_OSOBY
           WHERE o.ID_OSOBY = id_os;
       return v_ret;
   end rezerwacje_osoby;
/
```

c) przyszle_rezerwacje_osoby

```
CREATE OR REPLACE FUNCTION przyszle_rezerwacje_osoby(id_os INT)
   RETURN UCZESTNICY_WYCIECZKI_TABLE as v_ret UCZESTNICY_WYCIECZKI_TABLE;
       os_istnieje                              INTEGER;
   BEGIN
       SELECT COUNT(*) INTO os_istnieje FROM OSOBY
       WHERE OSOBY.ID_OSOBY = id_os;

       IF os_istnieje = 0 THEN
         raise_application_error(-20001, 'Brak osoby o podanym id');
       END IF;

       SELECT uczestnicy_wycieczki_row(w.ID_WYCIECZKI,  w.NAZWA, w.KRAJ,
w.DATA, o.IMIE,
                          o.NAZWISKO, r.STATUS)
       BULK COLLECT INTO v_ret
       FROM WYCIECZKI w
          JOIN REZERWACJE r ON w.ID_WYCIECZKI = r.ID_WYCIECZKI
          JOIN OSOBY o ON r.ID_OSOBY = o.ID_OSOBY
           WHERE o.ID_OSOBY = id_os AND CURRENT_DATE < w.DATA;
       return v_ret;
   end przyszle_rezerwacje_osoby;
```

d) dostepne_wycieczki

```
CREATE OR REPLACE FUNCTION DOSTEPNE_WYCIECZKI(kraj1  WYCIECZKI.KRAJ%TYPE,
data_od DATE, data_do DATE)
   RETURN WYCIECZKI_TABLE as v_ret WYCIECZKI_TABLE;
   BEGIN
       SELECT WYCIECZKI_ROW(w.ID_WYCIECZKI,  w.NAZWA, w.KRAJ, w.DATA,
                          w.LICZBA_MIEJSC, wm.LICZBA_WOLNYCH_MIEJSC)
       BULK COLLECT INTO v_ret
       FROM WYCIECZKI w
```

```
          JOIN WYCIECZKI_MIEJSCA WM on w.ID_WYCIECZKI = WM.ID_WYCIECZKI
           WHERE w.KRAJ = kraj1 AND data_od < w.DATA and data_do > w.DATA
AND wm.LICZBA_WOLNYCH_MIEJSC >0;
       return v_ret;
   end DOSTEPNE_WYCIECZKI;
```

3. Procedury modyfikujące dane

a) dodaj_rezerwacje

```
create PROCEDURE dodaj_rezerwacje(id_wyc INT, id_os INT)
AS
   istnieje_os INT;
   dostepne_wyc INT;
   istnieje    INT;
BEGIN
   SELECT COUNT(*) INTO istnieje_os FROM OSOBY o
   WHERE o.ID_OSOBY = id_os;

   SELECT COUNT(*) INTO dostepne_wyc
   FROM DOSTEPNE_WYCIECZKI_VIEW dw
   JOIN WYCIECZKI w ON w.ID_WYCIECZKI = dw.ID_WYCIECZKI
   WHERE dw.ID_WYCIECZKI = id_wyc AND dw.LICZBA_WOLNYCH_MIEJSC > 0 AND
w.DATA > CURRENT_DATE;

   SELECT COUNT(*) INTO istnieje FROM REZERWACJE r
   WHERE r.ID_WYCIECZKI = id_wyc AND r.ID_OSOBY = id_os;

   IF istnieje_os = 0 THEN
       raise_application_error(-102, 'Brak osoby o podanym id');
   end if;

   IF dostepne_wyc = 0 THEN
       raise_application_error(-103, 'Brak wycieczki o podanym id');
   end if;

   IF istnieje = 0 THEN
       raise_application_error(-104, 'Rezerwacja juz istnieje');
   end if;

   INSERT INTO REZERWACJE (ID_WYCIECZKI, ID_OSOBY, STATUS)
```

```
   VALUES (id_wyc, id_os, 'N');

end dodaj_rezerwacje;
```

b) zmien_status_rezerwacji

```
create PROCEDURE zmien_status_rezerwacji(id_rezerwacji INT, status
REZERWACJE.STATUS%TYPE)
AS
       dostepna INT;
       stary_status REZERWACJE.STATUS%TYPE;
       ilosc_wolnych INT;
BEGIN
   SELECT COUNT(*) INTO dostepna FROM REZERWACJE r
   WHERE r.NR_REZERWACJI = id_rezerwacji;

   IF dostepna = 0 THEN
       RAISE_APPLICATION_ERROR(-105, 'Brak rezerwacji o podanym id');
   end if;

   SELECT r.STATUS INTO stary_status
   FROM REZERWACJE r where r.NR_REZERWACJI = id_rezerwacji;

   IF stary_status = status THEN
       raise_application_error(-106, 'Status taki sam');
   end if;

   CASE
       WHEN status = 'N'
           THEN
           raise_application_error(-107, 'Rezerwacja nie moze byc nowa');

       WHEN stary_status = 'A'
           THEN
           Select dw.LICZBA_WOLNYCH_MIEJSC INTO ilosc_wolnych
           FROM DOSTEPNE_WYCIECZKI_VIEW dw
           JOIN Rezerwacje r ON r.ID_WYCIECZKI = dw.ID_WYCIECZKI
           WHERE id_rezerwacji = r.NR_REZERWACJI;

           IF ilosc_wolnych = 0 THEN
```

```
                raise_application_error(-108, 'Brak wolnych miejsc');
            end if;

        ELSE NULL;
        END CASE;

        UPDATE REZERWACJE
        SET STATUS = zmien_status_rezerwacji.status
        WHERE NR_REZERWACJI = zmien_status_rezerwacji.id_rezerwacji;
end zmien_status_rezerwacji;
```

c) zmien_liczbe_miejsc

```
CREATE PROCEDURE zmien_liczbe_miejsc(id_wycieczki
WYCIECZKI.ID_WYCIECZKI%TYPE, liczba_miejsc INT)
AS
   ilosc_zarezerwowanych INT;
   BEGIN
        SELECT COUNT(*)  INTO ilosc_zarezerwowanych FROM Wycieczki w
        LEFT JOIN Rezerwacje r ON r.ID_WYCIECZKI = w.ID_WYCIECZKI
        WHERE w.ID_WYCIECZKI = zmien_liczbe_miejsc.id_wycieczki
        GROUP BY w.ID_WYCIECZKI;

        IF ilosc_zarezerwowanych > liczba_miejsc THEN
            raise_application_error(-109, 'Nie mozna zmienic ilosci miejsc,
za duzo zarezerwowanych ');
        end if;

        UPDATE WYCIECZKI
        SET WYCIECZKI.LICZBA_MIEJSC = liczba_miejsc
        WHERE WYCIECZKI.ID_WYCIECZKI = id_wycieczki;
   end zmien_liczbe_miejsc;
```

4. Rezerwacje log - tabela dziennikująca rezerwacje

```
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_FK1 FOREIGN KEY
(
ID_REZERWACJI
   )
REFERENCES REZERWACJE
(
   NR_REZERWACJI
)
ENABLE;
```

Zmieniam procedury tak aby dopisywały informacje o rezerwacjach do rezerwacje_log.

a) dodaj_rezerwacje2

```
create PROCEDURE dodaj_rezerwacje2(id_wyc INT, id_os INT)
AS
   rezerwacja INT;
   istnieje_os INT;
   dostepne_wyc INT;
   istnieje    INT;
BEGIN
   SELECT COUNT(*) INTO istnieje_os FROM OSOBY o
   WHERE o.ID_OSOBY = id_os;

   SELECT COUNT(*) INTO dostepne_wyc
   FROM DOSTEPNE_WYCIECZKI_VIEW dw
   JOIN WYCIECZKI w ON w.ID_WYCIECZKI = dw.ID_WYCIECZKI
   WHERE dw.ID_WYCIECZKI = id_wyc AND dw.LICZBA_WOLNYCH_MIEJSC > 0 AND
w.DATA > CURRENT_DATE;

   SELECT COUNT(*) INTO istnieje FROM REZERWACJE r
   WHERE r.ID_WYCIECZKI = id_wyc AND r.ID_OSOBY = id_os;
```

```
   IF istnieje_os = 0 THEN
       raise_application_error(-102, 'Brak osoby o podanym id');
   end if;

   IF dostepne_wyc = 0 THEN
       raise_application_error(-103, 'Brak wycieczki o podanym id');
   end if;

   IF istnieje = 0 THEN
       raise_application_error(-104, 'Rezerwacja juz istnieje');
   end if;

   INSERT INTO REZERWACJE (ID_WYCIECZKI, ID_OSOBY, STATUS)
   VALUES (id_wyc, id_os, 'N');

   SELECT MAX(r.NR_REZERWACJI) INTO rezerwacja FROM REZERWACJE r;

   INSERT INTO REZERWACJE_LOG (ID_REZERWACJI, DATA, STATUS)
   VALUES (rezerwacja, CURRENT_DATE, 'N');

end dodaj_rezerwacje2;
/
```

b) Zmien_status_rezerwacji2

```
create PROCEDURE zmien_status_rezerwacji2(id_rezerwacji INT, status
REZERWACJE.STATUS%TYPE)
AS
       dostepna INT;
       stary_status REZERWACJE.STATUS%TYPE;
       ilosc_wolnych INT;
       data DATE;
BEGIN
   SELECT COUNT(*) INTO dostepna FROM REZERWACJE r
   WHERE r.NR_REZERWACJI = id_rezerwacji;

   IF dostepna = 0 THEN
       RAISE_APPLICATION_ERROR(-105, 'Brak rezerwacji o podanym id');
   end if;

   SELECT r.STATUS INTO stary_status
```

```
   FROM REZERWACJE r where r.NR_REZERWACJI = id_rezerwacji;

   IF stary_status = status THEN
       raise_application_error(-106, 'Status taki sam');
   end if;


   CASE
       WHEN status = 'N'
           THEN
           raise_application_error(-107, 'Rezerwacja nie moze byc nowa');

       WHEN stary_status = 'A'
           THEN
           Select dw.LICZBA_WOLNYCH_MIEJSC INTO ilosc_wolnych
           FROM DOSTEPNE_WYCIECZKI_VIEW dw
           JOIN Rezerwacje r ON r.ID_WYCIECZKI = dw.ID_WYCIECZKI
           WHERE id_rezerwacji = r.NR_REZERWACJI;

           IF ilosc_wolnych = 0 THEN
               raise_application_error(-108, 'Brak wolnych miejsc');
           end if;

       ELSE NULL;
       END CASE;

       UPDATE REZERWACJE
       SET STATUS =  zmien_status_rezerwacji2.status
       WHERE NR_REZERWACJI = zmien_status_rezerwacji2.id_rezerwacji;

       INSERT INTO REZERWACJE_LOG(ID_REZERWACJI, DATA, STATUS)
       VALUES (id_rezerwacji, CURRENT_DATE, status);
end zmien_status_rezerwacji2;
/
```

5. Dodanie pola liczba wolnych miejsc w tabeli wycieczki

```
ALTER TABLE WYCIECZKI
ADD liczba_wolnych_miejsc INT
```

```
CREATE PROCEDURE przelicz
AS
BEGIN
   UPDATE WYCIECZKI w
   SET liczba_wolnych_miejsc =
       LICZBA_MIEJSC -
              (SELECT COUNT(*) FROM REZERWACJE r
              where  w.ID_WYCIECZKI = r.ID_WYCIECZKI
              AND r.STATUS != 'A');
end przelicz;
```

Aktualizuję widoki

a) dostepne_wycieczki

```
CREATE VIEW DOSTEPNE_WYCIECZKI_VIEW2
AS
   SELECT
       w.ID_WYCIECZKI,
       w.KRAJ,
       w.NAZWA,
       w.LICZBA_MIEJSC,
       w.LICZBA_WOLNYCH_MIEJSC
   FROM WYCIECZKI w
   WHERE w.LICZBA_WOLNYCH_MIEJSC > 0 AND CURRENT_DATE > w.DATA;
```

b) Wycieczki_miejsca

```
CREATE VIEW WYCIECZKI_MIEJSCA2
AS
   SELECT
       w.ID_WYCIECZKI,
       w.KRAJ,
       w.DATA,
       w.NAZWA,
       w.LICZBA_MIEJSC,
       w.liczba_wolnych_miejsc
   FROM WYCIECZKI w;
```

```
CREATE PROCEDURE dodaj_rezerwacje3(id_wyc REZERWACJE.ID_WYCIECZKI%TYPE,
id_os REZERWACJE.ID_OSOBY%TYPE)
AS
   rezerwacja INT;
   istnieje_os INT;
   dostepne_wyc INT;
   istnieje INT;
BEGIN
   SELECT COUNT(*) INTO istnieje_os FROM OSOBY o
   WHERE o.ID_OSOBY = id_os;

   SELECT COUNT(*) INTO dostepne_wyc
   FROM WYCIECZKI w
   WHERE w.ID_WYCIECZKI = id_wyc AND w.LICZBA_WOLNYCH_MIEJSC > 0 AND w.DATA
> CURRENT_DATE;

   SELECT COUNT(*) INTO istnieje FROM REZERWACJE r
   WHERE r.ID_WYCIECZKI = id_wyc AND r.ID_OSOBY = id_os;

   IF istnieje_os = 0 THEN
       raise_application_error(-102, 'Brak osoby o podanym id');
   end if;

   IF dostepne_wyc = 0 THEN
       raise_application_error(-103, 'Wycieczka niedostepna');
   end if;

   IF istnieje = 0 THEN
       raise_application_error(-104, 'Rezerwacja juz istnieje');
   end if;


   INSERT INTO REZERWACJE(ID_WYCIECZKI, ID_OSOBY, STATUS)
   VALUES(id_wyc, id_os, 'N');

   SELECT MAX(r.NR_REZERWACJI) INTO rezerwacja FROM REZERWACJE r;

   INSERT INTO REZERWACJE_LOG (ID_REZERWACJI, DATA, STATUS)
   VALUES (rezerwacja, CURRENT_DATE, 'N');

   UPDATE WYCIECZKI w
   SET w.liczba_wolnych_miejsc = w.liczba_wolnych_miejsc -1
```

```
   WHERE w.ID_WYCIECZKI = id_wyc;
END dodaj_rezerwacje3;
```

c) zmien_status_rezerwacji3

```
create PROCEDURE zmien_status_rezerwacji3(id_rezerwacji INT, status
REZERWACJE.STATUS%TYPE)
AS
       dostepna INT;
       stary_status REZERWACJE.STATUS%TYPE;
       ilosc_wolnych INT;
       zmiana_miejsc INT;
       data DATE;
BEGIN
   SELECT COUNT(*) INTO dostepna FROM REZERWACJE r
   WHERE r.NR_REZERWACJI = id_rezerwacji;

   IF dostepna = 0 THEN
       RAISE_APPLICATION_ERROR(-105, 'Brak rezerwacji o podanym id');
   end if;

   SELECT r.STATUS INTO stary_status
   FROM REZERWACJE r where r.NR_REZERWACJI = id_rezerwacji;

   IF stary_status = status THEN
       raise_application_error(-106, 'Status taki sam');
   end if;

   CASE
       WHEN status = 'N'
           THEN
           raise_application_error(-107, 'Rezerwacja nie moze byc nowa');

       WHEN stary_status = 'A'
           THEN
           Select w.LICZBA_WOLNYCH_MIEJSC INTO ilosc_wolnych
           FROM WYCIECZKI w
           JOIN Rezerwacje r ON r.ID_WYCIECZKI = w.ID_WYCIECZKI
           WHERE id_rezerwacji = r.NR_REZERWACJI;
```

```
            IF ilosc_wolnych = 0 THEN
                raise_application_error(-108, 'Brak wolnych miejsc');
            ELSE
                zmiana_miejsc := 1;
            end if;

        ELSE-- tu sie pieprzy
           IF status = 'A'
            THEN
                zmiana_miejsc := -1;
            ELSE
               zmiana_miejsc := 0;
            end if;
        END CASE;

        UPDATE REZERWACJE
        SET STATUS =  zmien_status_rezerwacji3.status
        WHERE NR_REZERWACJI = zmien_status_rezerwacji3.id_rezerwacji;

        INSERT INTO REZERWACJE_LOG(ID_REZERWACJI, DATA, STATUS)
        VALUES (id_rezerwacji, CURRENT_DATE, status);

        UPDATE WYCIECZKI w
        SET w.liczba_wolnych_miejsc = w.liczba_wolnych_miejsc +
zmiana_miejsc;
end;
/
```

d) zmien_liczbe_miejsc

```
create PROCEDURE zmien_liczbe_miejsc2(id_wycieczki
WYCIECZKI.ID_WYCIECZKI%TYPE, liczba_miejsc INT)
AS
   ilosc_zarezerwowanych INT;
   BEGIN
        SELECT LICZBA_MIEJSC - LICZBA_WOLNYCH_MIEJSC INTO
ilosc_zarezerwowanych
        FROM WYCIECZKI w
        WHERE w.ID_WYCIECZKI = id_wycieczki;
        IF ilosc_zarezerwowanych > liczba_miejsc THEN
            raise_application_error(-109, 'Nie mozna zmienic ilosci miejsc,
za duzo zarezerwowanych ');
```

```
        end if;

        UPDATE WYCIECZKI w
        SET w.LICZBA_MIEJSC = liczba_miejsc,
            w.LICZBA_WOLNYCH_MIEJSC = w.LICZBA_WOLNYCH_MIEJSC -
                                      (w.LICZBA_MIEJSC - liczba_miejsc)
        WHERE w.ID_WYCIECZKI = id_wycieczki;


    end zmien_liczbe_miejsc2;
/
```

6. Triggery zapisywanie do dziennika rezerwacji

a) triggrer dodaj rezerwacje

```
CREATE OR REPLACE TRIGGER dodaj_rezerwacje_log
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;

create PROCEDURE DODAJ_REZERWACJE4(id_wyc REZERWACJE.ID_WYCIECZKI%TYPE,
id_os REZERWACJE.ID_OSOBY%TYPE)
AS
   rezerwacja INT;
   istnieje_os INT;
   dostepne_wyc INT;
   istnieje INT;
BEGIN
   SELECT COUNT(*) INTO istnieje_os FROM OSOBY o
   WHERE o.ID_OSOBY = id_os;

   SELECT COUNT(*) INTO dostepne_wyc
   FROM WYCIECZKI w
   WHERE w.ID_WYCIECZKI = id_wyc AND w.LICZBA_WOLNYCH_MIEJSC > 0 AND w.DATA
> CURRENT_DATE;

   SELECT COUNT(*) INTO istnieje FROM REZERWACJE r
   WHERE r.ID_WYCIECZKI = id_wyc AND r.ID_OSOBY = id_os;
```

```
   IF istnieje_os = 0 THEN
       raise_application_error(-102, 'Brak osoby o podanym id');
   end if;

   IF dostepne_wyc = 0 THEN
       raise_application_error(-103, 'Wycieczka niedostepna');
   end if;

   IF istnieje = 0 THEN
       raise_application_error(-104, 'Rezerwacja juz istnieje');
   end if;

   INSERT INTO REZERWACJE(ID_WYCIECZKI, ID_OSOBY, STATUS)
   VALUES(id_wyc, id_os, 'N');

   SELECT MAX(r.NR_REZERWACJI) INTO rezerwacja FROM REZERWACJE r;

   UPDATE WYCIECZKI w
   SET w.liczba_wolnych_miejsc = w.liczba_wolnych_miejsc -1
   WHERE w.ID_WYCIECZKI = id_wyc;
END DODAJ_REZERWACJE4;
```

b) trigger zmiana statusu rezerwacji

```
CREATE TRIGGER zmiana_statusu_rezerwacji_log
AFTER UPDATE OF STATUS ON REZERWACJE FOR EACH ROW
BEGIN
   INSERT INTO REZERWACJE_LOG(ID_REZERWACJI, DATA, STATUS)
   VALUES(:NEW.NR_REZERWACJI,CURRENT_DATE,:NEW.STATUS);
End;
```

c) zmiana procedury zmien_status_rezerwacji

```
create PROCEDURE zmien_status_rezerwacji4(id_rezerwacji INT, status
REZERWACJE.STATUS%TYPE)
AS
       dostepna INT;
       stary_status REZERWACJE.STATUS%TYPE;
```

```
       ilosc_wolnych INT;
       zmiana_miejsc INT;
       data DATE;
BEGIN
   SELECT COUNT(*) INTO dostepna FROM REZERWACJE r
   WHERE r.NR_REZERWACJI = id_rezerwacji;

   IF dostepna = 0 THEN
       RAISE_APPLICATION_ERROR(-105, 'Brak rezerwacji o podanym id');
   end if;

   SELECT r.STATUS INTO stary_status
   FROM REZERWACJE r where r.NR_REZERWACJI = id_rezerwacji;

   IF stary_status = status THEN
       raise_application_error(-106, 'Status taki sam');
   end if;

   CASE
       WHEN status = 'N'
           THEN
           raise_application_error(-107, 'Rezerwacja nie moze byc nowa');

       WHEN stary_status = 'A'
           THEN
           Select w.LICZBA_WOLNYCH_MIEJSC INTO ilosc_wolnych
           FROM WYCIECZKI w
           JOIN Rezerwacje r ON r.ID_WYCIECZKI = w.ID_WYCIECZKI
           WHERE id_rezerwacji = r.NR_REZERWACJI;

           IF ilosc_wolnych = 0 THEN
               raise_application_error(-108, 'Brak wolnych miejsc');
           ELSE
               zmiana_miejsc := 1;
           end if;

       ELSE
          IF status = 'A'
           THEN
               zmiana_miejsc := -1;
           ELSE
              zmiana_miejsc := 0;
```

```
            end if;
        END CASE;

        UPDATE REZERWACJE
        SET STATUS =  zmien_status_rezerwacji4.status
        WHERE NR_REZERWACJI = zmien_status_rezerwacji4.id_rezerwacji;

        UPDATE WYCIECZKI w
        SET w.liczba_wolnych_miejsc = w.liczba_wolnych_miejsc +
zmiana_miejsc;
end;
/
```

d) trigger zabraniajacy usuniecia rezerwacji

```
CREATE TRIGGER zabron_usuniecia_rezerwacji
BEFORE DELETE ON REZERWACJE FOR EACH ROW
BEGIN
   RAISE_APPLICATION_ERROR(-111, 'Nie mozna usunac tej rezerwacji', TRUE);
END;
```

7. Obsługa pola liczba wolnych miejsc

a) trigger dodanie nowej rezerwacji

```
CREATE OR REPLACE TRIGGER dodaj_rezerwacje_log
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 w.liczba_wolnych_miejsc = w.liczba_wolnych_miejsc - 1;
END;
```

b) zmieniona procedura dodaj_rezerwacje

```
create PROCEDURE DODAJ_REZERWACJE4(id_wyc REZERWACJE.ID_WYCIECZKI%TYPE,
id_os REZERWACJE.ID_OSOBY%TYPE)
AS
   rezerwacja INT;
```

```
   istnieje_os INT;
   dostepne_wyc INT;
   istnieje INT;
BEGIN
   SELECT COUNT(*) INTO istnieje_os FROM OSOBY o
   WHERE o.ID_OSOBY = id_os;

   SELECT COUNT(*) INTO dostepne_wyc
   FROM WYCIECZKI w
   WHERE w.ID_WYCIECZKI = id_wyc AND w.LICZBA_WOLNYCH_MIEJSC > 0 AND w.DATA
> CURRENT_DATE;

   SELECT COUNT(*) INTO istnieje FROM REZERWACJE r
   WHERE r.ID_WYCIECZKI = id_wyc AND r.ID_OSOBY = id_os;

   IF istnieje_os = 0 THEN
       raise_application_error(-102, 'Brak osoby o podanym id');
   end if;

   IF dostepne_wyc = 0 THEN
       raise_application_error(-103, 'Wycieczka niedostepna');
   end if;

   IF istnieje = 0 THEN
       raise_application_error(-104, 'Rezerwacja juz istnieje');
   end if;

   INSERT INTO REZERWACJE(ID_WYCIECZKI, ID_OSOBY, STATUS)
   VALUES(id_wyc, id_os, 'N');
END DODAJ_REZERWACJE4;
/
```

c) Trigger zmiana_statusu_rezerwacji

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

   IF :OLD.STATUS = 'A' AND :NEW.STATUS != 'A' THEN
```

```
        zmiana := 1;
    end if;

    IF :OLD.STATUS != 'A' AND :NEW.STATUS = 'A' THEN
        zmiana := -1;
    ELSE     zmiana := 0;
    end if;

    UPDATE WYCIECZKI w
    SET LICZBA_WOLNYCH_MIEJSC = LICZBA_WOLNYCH_MIEJSC + zmiana
    WHERE w.ID_WYCIECZKI = :NEW.ID_WYCIECZKI;
end;
```

d) zmiana procedury zmiana_statusu_rezerwacji

```
create PROCEDURE zmien_status_rezerwacji4(id_rezerwacji INT, status
REZERWACJE.STATUS%TYPE)
AS
        dostepna INT;
        stary_status REZERWACJE.STATUS%TYPE;
        ilosc_wolnych INT;
        zmiana_miejsc INT;
BEGIN
    SELECT COUNT(*) INTO dostepna FROM REZERWACJE r
    WHERE r.NR_REZERWACJI = id_rezerwacji;

    IF dostepna = 0 THEN
        RAISE_APPLICATION_ERROR(-105, 'Brak rezerwacji o podanym id');
    end if;

    SELECT r.STATUS INTO stary_status
    FROM REZERWACJE r where r.NR_REZERWACJI = id_rezerwacji;

    IF stary_status = status THEN
        raise_application_error(-106, 'Status taki sam');
    end if;

    CASE
        WHEN status = 'N'
            THEN
            raise_application_error(-107, 'Rezerwacja nie moze byc nowa');
```

```
       WHEN stary_status = 'A'
           THEN
           Select w.LICZBA_WOLNYCH_MIEJSC INTO ilosc_wolnych
           FROM WYCIECZKI w
           JOIN Rezerwacje r ON r.ID_WYCIECZKI = w.ID_WYCIECZKI
           WHERE id_rezerwacji = r.NR_REZERWACJI;

           IF ilosc_wolnych = 0 THEN
               raise_application_error(-108, 'Brak wolnych miejsc');
           end if;

           ELSE NULL;
       END CASE;

       UPDATE REZERWACJE
       SET STATUS =  zmien_status_rezerwacji4.status
       WHERE NR_REZERWACJI = zmien_status_rezerwacji4.id_rezerwacji;
end;
/
```

e) trigger zmiana_liczby_miejsc

```
create or replace TRIGGER  OBSLUGUJ_ZMIANA_LICZBY_MIEJSC
BEFORE UPDATE OF LICZBA_MIEJSC ON WYCIECZKI FOR EACH ROW
DECLARE
   ilosc number ;
BEGIN
   SELECT count ( * ) INTO ilosc
   FROM REZERWACJE r WHERE r. ID_WYCIECZKI = :NEW.ID_WYCIECZKI
                       AND r. STATUS != 'A' ;
   :NEW.LICZBA_WOLNYCH_MIEJSC := (:new.LICZBA_MIEJSC - ilosc) ;
END ;
```

f) zmiana procedury zmiana_liczby_miejsc

```
create PROCEDURE zmien_liczbe_miejsc3(id_wycieczki
WYCIECZKI.ID_WYCIECZKI%TYPE, liczba_miejsc INT)
AS
   ilosc_zarezerwowanych INT;
   BEGIN
       SELECT LICZBA_MIEJSC - LICZBA_WOLNYCH_MIEJSC INTO
```

```
ilosc_zarezerwowanych
       FROM WYCIECZKI w
       WHERE w.ID_WYCIECZKI = id_wycieczki;
       IF ilosc_zarezerwowanych > liczba_miejsc THEN
           raise_application_error(-109, 'Nie mozna zmienic ilosci miejsc,
za duzo zarezerwowanych ');
       end if;

       UPDATE WYCIECZKI w
       SET w.LICZBA_MIEJSC = liczba_miejsc
       WHERE w.ID_WYCIECZKI = id_wycieczki;
   end zmien_liczbe_miejsc3;
/
```