Sprawozdanie z laboratorium 1

3.WIDOKI

-widok wyświetlający wycieczki danej osoby

```
CREATE VIEW WYCIECZKI_OSOBY 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
```

-widok wyświetlający potwierdzone wycieczki

```
CREATE VIEW WYCIECZKI_OSOBY_POTWIERDZONE AS SELECT
 w.ID_WYCIECZKI,
 w.NAZWA,
 w.KRAJ,
 w.DATA,
 o.IMIE,
 o.NAZWISKO
 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'
```

-widok wyświetlający przyszłe wycieczki osoby

```
CREATE VIEW WYCIECZKI_OSOBY_PRZYSZLE AS SELECT
  w.ID_WYCIECZKI,
  w.NAZWA,
  w.KRAJ,
  w.DATA,
  o.IMIE,
  o.NAZWISKO
  from WYCIECZKI w
```

```
  JOIN REZERWACJE r ON w.ID_WYCIECZKI = r.ID_WYCIECZKI
  JOIN OSOBY o ON r.ID_OSOBY = o.ID_OSOBY
  WHERE TO_DATE(SYSDATE,'YYYY-MM-DD') < TO_DATE(w.DATA,
'YYYY-MM-DD')
```

-widok wyświetlający ilość wolnych miejsc na daną wycieczkę

```
CREATE VIEW WYCIECZKI_MIEJSCA_WOLNE AS SELECT
   w.ID_WYCIECZKI,
   w.NAZWA,
   w.KRAJ,
   w.DATA,
   w.LICZBA_MIEJSC - (SELECT COUNT (*) from REZERWACJE r where
r.ID_wYCIECZKI = w.ID_WYCIECZKI AND r.STATUS != 'A' )
         AS  "wolne miejsca"
   from WYCIECZKI w
I
/
```

-widok wyświetlający anulowane wycieczki

```
CREATE VIEW WYCIECZKI_OSOBY_ANULOWANE AS SELECT
 w.ID_WYCIECZKI,
 w.NAZWA,
 w.KRAJ,
 w.DATA,
 o.IMIE,
 o.NAZWISKO
 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 = 'A'
```

-widok wyświetlający dostępne wycieczki

```
CREATE VIEW WYCIECZKI_DOSTEPNE AS SELECT
    w.ID_WYCIECZKI,
```

```
    w.NAZWA,
    w.KRAJ,
   w.DATA
  from WYCIECZKI w
   JOIN REZERWACJE r ON w.ID_WYCIECZKI = r.ID_WYCIECZKI
  WHERE re.STATUS != 'A' AND (SELECT COUNT (*) from REZERWACJE
re  where re.ID_WYCIECZKI = w.ID_WYCIECZKI) < w.liczba_miejsc
and w.DATA > current_date

/
```

-widok wyświetlający rezerwacje do anulowania

```
CREATE OR REPLACE VIEW WYCIECZKI_DO_ANULOWANIA AS SELECT
    r.NR_REZERWACJI,
    w.ID_WYCIECZKI,
    w.NAZWA,
    w.KRAJ,
    w.DATA
    from WYCIECZKI w
    JOIN REZERWACJE r ON w.ID_WYCIECZKI = r.ID_WYCIECZKI
    WHERE w.DATA >= (CURRENT_DATE + 7)
```

widok wyświetlający wycieczki, w których ponad połowa miejsc jest zarezerwowana

```
CREATE VIEW WYCIECZKI_PONAD_POLOWA AS SELECT
    w.ID_WYCIECZKI,
    w.NAZWA,
    w.KRAJ,
   w.DATA
  from WYCIECZKI w
   JOIN REZERWACJE r ON w.ID_WYCIECZKI = r.ID_WYCIECZKI
  WHERE r.STATUS != 'A'  AND (SELECT COUNT (*) from REZERWACJE
re  where re.ID_WYCIECZKI = w.ID_WYCIECZKI) BETWEEN
(w.LICZBA_WOLNYCH_MIEJSC * 0.5
  and   w.LICZBA_WOLNYCH_MIEJSC)    and w.DATA > current_date
/
```

4.FUNKCJE ZWRACAJĄCE TABELE

Tutaj należało zdefiniować odpowiednie typy zwracanych tabel.

-uczestnicy danej 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
UCZSETNICY_WYCIECZKI_ROW
/
```

```
CREATE OR REPLACE FUNCTION uczestnicy_wycieczki
(id_wycieczki_id IN INT)
 RETURN UCZESTNICY_WYCIECZKI_TABLE PIPELINED IS
 curr SYS_REFCURSOR;
 result_row UCZSETNICY_WYCIECZKI_ROW
            :=
UCZESTNICY_WYCIECZKI_ROW(NULL,NULL,NULL,NULL,NULL,NULL,NULL);
BEGIN
 OPEN curr FOR
   SELECT * FROM WYCIECZKI_OSOBY WHERE ID_WYCIECZKI=
id_wycieczki_id;
 LOOP
   FETCH curr INTO result_row.ID_WYCIECZKI, result_row.nazwa,
result_row.KRAJ, result_row.DATA, result_row.IMIE,
                   result_row.NAZWISKO, result_row.STATUS;
   EXIT WHEN curr%NOTFOUND;
   PIPE ROW(result_row);
 END LOOP;
```

```
 CLOSE curr;
 RETURN;
END;
/
```

-rezerwacje osoby

```
CREATE TYPE uczestnicy_wycieczki_row2 AS OBJECT (
 ID_WYCIECZKI INT,
 ID_rezerwacji INT,
 NAZWA VARCHAR2(100),
 KRAJ VARCHAR2(50),
 DATA DATE,
 IMIE VARCHAR2(50),
 NAZWISKO VARCHAR2(50),
 STATUS CHAR(1)
)

CREATE TYPE UCZESTNICY_WYCIECZKI_TABLE2 AS TABLE OF
UCZESTNICY_WYCIECZKI_ROW2


CREATE OR REPLACE FUNCTION wycieczki_danej_osoby(id_osoby_id
IN INT)
 RETURN UCZESTNICY_WYCIECZKI_TABLE2 PIPELINED IS
 curr SYS_REFCURSOR;
 result_row uczestnicy_wycieczki_row2 :=
uczestnicy_wycieczki_row2(NULL,NULL,NULL,NULL,NULL,NULL,NULL,N
ULL);
 BEGIN
   OPEN curr FOR
     SELECT w.ID_WYCIECZKI,r.NR_REZERWACJI, w.NAZWA, w.KRAJ,
w.data,o.imie,o.NAZWISKO, r.status
     from wycieczki w
     inner join rezerwacje r on w.id_wycieczki =
r.id_wycieczki
     inner join osoby o on o.id_osoby = r.id_osoby
     where r.id_osoby = id_osoby_id;
```

```
   LOOP
        FETCH curr INTO result_row.ID_WYCIECZKI,
result_row.id_rezerwacji, result_row.nazwa,result_row.KRAJ,
result_row.DATA, result_row.IMIE,
                        result_row.NAZWISKO, result_row.STATUS;
        EXIT WHEN curr%NOTFOUND;
        PIPE ROW(result_row);
   END LOOP;
   CLOSE curr;
   RETURN;
 END;
```

-przyszłe rezerwacje osoby

```
CREATE TYPE uczestnicy_wycieczki_row3 AS OBJECT (
 ID_WYCIECZKI INT,
 ID_rezerwacji INT,
 NAZWA VARCHAR2(100),
 KRAJ VARCHAR2(50),
 DATA DATE,
 IMIE VARCHAR2(50),
 NAZWISKO VARCHAR2(50),
 STATUS CHAR(1)
)
CREATE TYPE UCZESTNICY_WYCIECZKI_TABLE3 AS TABLE OF
UCZESTNICY_WYCIECZKI_ROW3


CREATE OR REPLACE FUNCTION
przyszle_rezerwacje_osoby(id_osoby_id IN INT)
 RETURN UCZESTNICY_WYCIECZKI_TABLE3 PIPELINED  IS
 curr SYS_REFCURSOR;
 result_row uczestnicy_wycieczki_row3 :=
uczestnicy_wycieczki_row3(NULL,NULL,NULL,NULL,NULL,NULL,NULL,N
ULL);
 BEGIN
   OPEN curr FOR
   SELECT w.ID_WYCIECZKI, r.nr_rezerwacji, w.NAZWA, w.KRAJ,
w.data,o.imie,o.NAZWISKO, r.status
   from wycieczki w
   inner join rezerwacje r on w.id_wycieczki = r.id_wycieczki
   inner join osoby o on o.id_osoby = r.id_osoby
```

```
   where r.id_osoby = id_osoby_id and w.DATA > CURRENT_DATE;

 LOOP
     FETCH curr INTO result_row.ID_WYCIECZKI,
result_row.id_rezerwacji, result_row.nazwa,result_row.KRAJ,
result_row.DATA, result_row.IMIE,
                        result_row.NAZWISKO, result_row.STATUS;
     EXIT WHEN curr%NOTFOUND;
     PIPE ROW(result_row);
 END LOOP;
 CLOSE curr;
 RETURN;
END;
```

-dostępne 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 WYCIECZKI_TABLE AS TABLE OF WYCIECZKI_ROW


CREATE OR REPLACE FUNCTION dostepne_wycieczki2
RETURN wycieczki_table  PIPELINED IS
 curr SYS_REFCURSOR;
 result_row wycieczki_row :=
wycieczki_row(NULL,NULL,NULL,NULL);
 BEGIN
   OPEN curr FOR
   SELECT w.ID_WYCIECZKI, w.NAZWA, w.KRAJ, w.DATA
   from wycieczki w
   WHERE w.DATA > current_date and w.LICZBA_WOLNYCH_MIEJSC >
0;

 LOOP
```

```
    FETCH curr INTO result_row.ID_WYCIECZKI,
result_row.nazwa, result_row.kraj, result_row.data;
    EXIT WHEN curr%NOTFOUND;
    PIPE ROW(result_row);
 END LOOP;
 CLOSE curr;
 RETURN;
END;
/
```

5.PROCEDURY MODYFIKUJĄCE DANE

-dodaj rezerwacje

```
CREATE OR REPLACE PROCEDURE dodaj_rezerwacje(id_wycieczki_id
IN INT, id_osoby_id IN INT) AS
osoba  INT;
wycieczka  INT;
ilosc  INT;
limit  INT;

BEGIN
 BEGIN
   SELECT   count(osoby.id_osoby)  into  osoba FROM  osoby
where OSOBY.ID_OSOBY = id_osoby_id;
   SELECT  count(wycieczki.id_wycieczki) into wycieczka from
WYCIECZKI where WYCIECZKI.ID_WYCIECZKI = id_wycieczki_id;
   select count(*) into ilosc  from rezerwacje  where
REZERWACJE.id_wycieczki = id_wycieczki_id and
REZERWACJE.STATUS != 'A';
   SELECT wycieczki.LICZBA_MIEJSC into limit from WYCIECZKI
where WYCIECZKI.ID_WYCIECZKI = id_wycieczki_id;
 END;
 if(osoba = 0)
 THEN
   raise_application_error (-20001, 'Nie ma osoby o takim  id
' , TRUE);
 END IF ;
 if(wycieczka = 0 )
 THEN
```

```
   raise_application_error (-20001, 'Nie ma wycieczki o takim
id' , TRUE);
 END IF;
 if(ilosc + 1 <= limit)
 THEN
   INSERT INTO rezerwacje
   (id_wycieczki, id_osoby, status)
   VALUES(id_wycieczki_id, id_osoby_id, 'N');
else
      raise_application_error (-20001, 'Nie ma wolnych miejsc '
, TRUE);
 end IF;
 RETURN;
END;
```

-zmień status rezerwacji

```
CREATE OR REPLACE PROCEDURE
zmien_status_rezerwacji(id_rezerwacji_id IN INT, status IN
char)
AS
rezerwacja INT;
stan char(1);
BEGIN
 BEGIN
   SELECT count(rezerwacje.nr_rezerwacji) into rezerwacja from
REZERWACJE where rezerwacje.NR_REZERWACJI = id_rezerwacji_id;
   SELECT rezerwacje.STATUS into stan from REZERWACJE where
REZERWACJE.NR_REZERWACJI = id_rezerwacji_id;
 end;
if (rezerwacja = 0)
 then
   raise_application_error (-20001, 'Nie ma wycieczki o takim
id ' , TRUE);
 end if;
 if stan = 'A'
 then
```

```
   raise_application_error (-20001, 'Wycieczka jest anulowana
' , TRUE);
 end if;
 UPDATE rezerwacje
 SET status = 'P'
 WHERE nr_rezerwacji = id_rezerwacji_id;
return ;
END;
/
```

-zmień ilość miejsc

```
CREATE OR REPLACE  PROCEDURE
zmien_ilosc_miejsc(id_wycieczki_id IN INT, ilosc_miejsc IN
INT)
AS
wycieczka INT;
ilosc INT;
ilosc_wolnych_miejsc INT;
BEGIN
 BEGIN
   select count(wycieczki.ID_WYCIECZKI) into wycieczka from
WYCIECZKI where WYCIECZKI.ID_WYCIECZKI = id_wycieczki_id;
   SELECT  count(*) into ilosc from rezerwacje r where
r.id_wycieczki = id_wycieczki_id;
   select wycieczki.liczba_miejsc into ilosc_wolnych_miejsc
from WYCIECZKI where WYCIECZKI.ID_WYCIECZKI = id_wycieczki_id;
 end;
 if(wycieczka = 0 )
   THEN
   raise_application_error (-20001, 'Nie ma wycieczki o takim
id  ' , TRUE);
 END IF;

 if(ilosc < ilosc_wolnych_miejsc )
 then
   update wycieczki
   SET LICZBA_MIEJSC = ilosc_miejsc
   where id_wycieczki = id_wycieczki_id;
else
```

```
  raise_application_error (-20001, 'Nie ma wolnych miejsc ' ,
TRUE);
 end if;
 return;
end;
```

6.REZERWACJE_LOG - NOWA TABELA

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

ALTER TABLE REZERWACJE_LOG
ADD CONSTRAINT REZERWACJE_LOG_FK1 FOREIGN KEY
(
NR_REZERWACJI
)
REFERENCES REZERWACJE
(
NR_REZERWACJI
)
ENABLE;
```

zmiana procedury dodającej rezerwacje:

```
CREATE OR REPLACE  PROCEDURE dodaj_rezerwacje2(id_wycieczki_id
IN INT, id_osoby_id IN INT) AS
osoba  INT;
wycieczka  INT;
ile_wolnych INT;
```

```
ostatnia_rezerwacja INT;

BEGIN
 BEGIN
   SELECT  count( osoby.id_osoby)  into  osoba FROM  osoby
where OSOBY.ID_OSOBY = id_osoby_id;
   SELECT  count(wycieczki.id_wycieczki) into wycieczka from
WYCIECZKI where WYCIECZKI.ID_WYCIECZKI = id_wycieczki_id;
   SELECT  wycieczki.LICZBA_WOLNYCH_MIEJSC into ile_wolnych
from wycieczki where WYCIECZKI.ID_WYCIECZKI = id_wycieczki_id;
 END;
 if(osoba = 0 )
 THEN
   raise_application_error (-20001, 'Nie ma osoby o takim  id
' , TRUE);
 END IF ;
 if(wycieczka  = 0 )
 THEN
   raise_application_error (-20001, 'Nie ma wyciezcki o takim
id  ' , TRUE);
 END IF;
 if(ile_wolnych > 0)
 THEN
   update WYCIECZKI
   set LICZBA_WOLNYCH_MIEJSC = LICZBA_WOLNYCH_MIEJSC -1
   where WYCIECZKI.ID_WYCIECZKI = wycieczka;
   INSERT INTO rezerwacje
   (id_wycieczki, id_osoby, status)
   VALUES( id_wycieczki_id, id_osoby_id, 'N');
   BEGIN
     SELECT max(REZERWACJE.NR_REZERWACJI) into
ostatnia_rezerwacja from REZERWACJE;
   END;
   INSERT  INTO REZERWACJE_LOG
   (NR_REZERWACJI,STATUS, DATA)
    values(ostatnia_rezerwacja, 'N',current_date ) ;
else
      raise_application_error (-20001, 'Nie ma wolnych miejsc
      ' , TRUE);
 end IF;
 RETURN;
END;
```

```
/
```

7.DODANIE W TABELI WYCIECZKI POLA LICZBA_WOLNYCH_MIEJSC

```
alter TABLE WYCIECZKI
ADD(LICZBA_WOLNYCH_MIEJSC INT)
```

-zmiana widoków:

```
CREATE VIEW WYCIECZKI_DOSTEPNE2 AS SELECT
    w.ID_WYCIECZKI,
    w.NAZWA,
    w.KRAJ,
   w.DATA,
   w.LICZBA_WOLNYCH_MIEJSC
  from WYCIECZKI w
   JOIN REZERWACJE r ON w.ID_WYCIECZKI = r.ID_WYCIECZKI
  WHERE r.STATUS != 'A' AND (SELECT COUNT (*) from REZERWACJE
re  where re.ID_WYCIECZKI = w.ID_WYCIECZKI) < w.liczba_miejsc
and w.DATA > current_date
/

CREATE VIEW WYCIECZKI_PONAD_POLOWA2 AS SELECT
    w.ID_WYCIECZKI,
    w.NAZWA,
    w.KRAJ,
   w.DATA
  from WYCIECZKI w
   JOIN REZERWACJE r ON w.ID_WYCIECZKI = r.ID_WYCIECZKI
  WHERE r.STATUS != 'A'  AND w.LICZBA_WOLNYCH_MIEJSC BETWEEN
(w.LICZBA_WOLNYCH_MIEJSC * 0.5
  and   w.LICZBA_WOLNYCH_MIEJSC)    and w.DATA > current_date
/
```

-procedura dodawania rezerwacji z aktualizacją liczby wolnych miejsc

```
CREATE OR REPLACE  PROCEDURE dodaj_rezerwacje2(id_wycieczki_id
IN INT, id_osoby_id IN INT) AS
osoba  INT;
wycieczka  INT;
ile_wolnych INT;
ostatnia_rezerwacja INT;
```

```
BEGIN
 BEGIN
   SELECT  count( osoby.id_osoby)  into  osoba FROM  osoby
where OSOBY.ID_OSOBY = id_osoby_id;
   SELECT  count(wycieczki.id_wycieczki) into wycieczka from
WYCIECZKI where WYCIECZKI.ID_WYCIECZKI = id_wycieczki_id;
   SELECT  wycieczki.LICZBA_WOLNYCH_MIEJSC into ile_wolnych
from wycieczki where WYCIECZKI.ID_WYCIECZKI = id_wycieczki_id;
 END;
 if(osoba = 0 )
 THEN
   raise_application_error (-20001, 'Nie ma osoby o takim  id
' , TRUE);
 END IF ;
 if(wycieczka  = 0 )
 THEN
   raise_application_error (-20001, 'Nie ma wyciezcki o takim
id  ' , TRUE);
 END IF;
 if(ile_wolnych > 0)
 THEN
   update WYCIECZKI
   set LICZBA_WOLNYCH_MIEJSC = LICZBA_WOLNYCH_MIEJSC -1
   where WYCIECZKI.ID_WYCIECZKI = wycieczka;
   INSERT INTO rezerwacje
   (id_wycieczki, id_osoby, status)
   VALUES( id_wycieczki_id, id_osoby_id, 'N');
   BEGIN
     SELECT max(REZERWACJE.NR_REZERWACJI) into
ostatnia_rezerwacja from REZERWACJE;
   END;
   INSERT  INTO REZERWACJE_LOG
   (NR_REZERWACJI,STATUS, DATA)
    values(ostatnia_rezerwacja, 'N',current_date ) ;
else
     raise_application_error (-20001, 'Nie ma wolnych miejsc '
, TRUE);
 end IF;
 RETURN;
END;
/
```

-procedura aktualizująca liczbę wolnych miejsc dla już istniejących danych:

```
CREATE PROCEDURE zaktualizuj_wolne_miejsca IS
   cur SYS_REFCURSOR;
   wycieczka NUMERIC;
   zajete INT;
 BEGIN
   OPEN cur FOR
     SELECT w.ID_WYCIECZKI, COUNT(w.ID_WYCIECZKI)
     FROM WYCIECZKI w
       LEFT JOIN REZERWACJE r  ON r.ID_WYCIECZKI =
w.ID_WYCIECZKI
       where r.STATUS !='A'
     GROUP BY w.ID_WYCIECZKI, LICZBA_MIEJSC;

   LOOP
     FETCH cur INTO wycieczka, zajete;
     EXIT WHEN cur%NOTFOUND;

     UPDATE WYCIECZKI
       SET LICZBA_WOLNYCH_MIEJSC = LICZBA_MIEJSC - zajete
       WHERE ID_WYCIECZKI = wycieczka;

   END LOOP;
   CLOSE cur;
 END;
/
```

-zmiana funkcji pokazującej dostępne wycieczki

```
CREATE FUNCTION dostepne_wycieczki2
RETURN wycieczki_table  PIPELINED IS
 curr SYS_REFCURSOR;
 result_row wycieczki_row :=
wycieczki_row(NULL,NULL,NULL,NULL);
 BEGIN
   OPEN curr FOR
   SELECT w.ID_WYCIECZKI, w.NAZWA, w.KRAJ, w.DATA
   from wycieczki w
```

```
   WHERE w.DATA > current_date and w.LICZBA_WOLNYCH_MIEJSC >
0;

 LOOP
     FETCH curr INTO result_row.ID_WYCIECZKI,
result_row.nazwa, result_row.kraj, result_row.data;
     EXIT WHEN curr%NOTFOUND;
     PIPE ROW(result_row);
 END LOOP;
 CLOSE curr;
 RETURN;
END;
/
```

8.ZAPISYWANIE DO DZIENNICZKA REZERWACJI - TRIGGERY.

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

-zmiana procedury dodającej rezerwacje

```
CREATE OR REPLACE PROCEDURE dodaj_rezerwacje4(id_wycieczki_id
IN INT, id_osoby_id IN INT) AS
osoba  INT;
wycieczka  INT;
ile_wolnych INT;
ostatnia_rezerwacja INT;

BEGIN
 BEGIN
   SELECT   count(osoby.id_osoby)  into  osoba FROM  osoby
where OSOBY.ID_OSOBY = id_osoby_id;
   SELECT  count(wycieczki.id_wycieczki) into wycieczka from
WYCIECZKI where WYCIECZKI.ID_WYCIECZKI = id_wycieczki_id;
   SELECT  wycieczki.LICZBA_WOLNYCH_MIEJSC into ile_wolnych
from wycieczki where WYCIECZKI.ID_WYCIECZKI = id_wycieczki_id;
```

```
 END;
 if(osoba = 0 )
 THEN
    raise_application_error (-20001, 'Nie ma osoby o takim  id
' , TRUE);
 END IF ;
 if(wycieczka = 0 )
 THEN
    raise_application_error (-20001, 'Nie ma wycieczki o takim
id  ' , TRUE);
 END IF;
 if(ile_wolnych > 0)
 THEN
    INSERT INTO rezerwacje
    (id_wycieczki, id_osoby, status)
    VALUES( id_wycieczki_id, id_osoby_id, 'N');
    BEGIN
       SELECT max(REZERWACJE.NR_REZERWACJI) into
ostatnia_rezerwacja from REZERWACJE;
    END;
else
       raise_application_error (-20001, 'Nie ma wolnych miejsc '
, TRUE);
 end IF;
 RETURN;
END;
```

- zmiana statusu rezerwacji

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

-zmiana procedury zmieniającej status rezerwacji

```
 CREATE OR REPLACE PROCEDURE
zmien_status_rezerwacji3(id_rezerwacji_id IN INT, status IN
char)
```

```
AS
rezerwacja INT;
stan char(1);
BEGIN
 BEGIN
   SELECT count(rezerwacje.nr_rezerwacji) into rezerwacja from
REZERWACJE where rezerwacje.NR_REZERWACJI = id_rezerwacji_id;
   SELECT rezerwacje.STATUS into stan from REZERWACJE where
REZERWACJE.NR_REZERWACJI = id_rezerwacji_id;
 end;
if (rezerwacja = 0)
 then
   raise_application_error (-20001, 'Nie ma wycieczki o takim
id ' , TRUE);
 end if;
 if stan = 'A'
 then
   raise_application_error (-20001, 'Wycieczka anulowana' ,
TRUE);
 end if;
 UPDATE rezerwacje
 SET status = 'P'
 WHERE nr_rezerwacji = id_rezerwacji_id;
return ;
END;
```

-trigger zabraniający usuwania wycieczki

```
 CREATE OR REPLACE TRIGGER blokuj_usuwanie
BEFORE DELETE ON REZERWACJE FOR EACH ROW
 BEGIN
   RAISE_APPLICATION_ERROR(-20000, 'Nie można usunąć tej
rezerwacji', TRUE);
 END;
```

9.TRIGGERY OBSŁUGUJĄCE POLE LICZBA_WOLNYCH_MIEJSC W TABELI WYCIECZKI

-aktualizacja liczby wolnych miejsc przy dodawaniu rezerwacji

```
create or replace TRIGGER zaktualizuj_wolne_miejsca
```

```
 AFTER INSERT on REZERWACJE for EACH ROW
 BEGIN
   update WYCIECZKI
     set LICZBA_WOLNYCH_MIEJSC = LICZBA_WOLNYCH_MIEJSC -1
   where WYCIECZKI.ID_WYCIECZKI = :NEW.ID_WYCIECZKI;

 END;
```

-zmiana procedury dodającej rezerwacje

```
CREATE OR REPLACE PROCEDURE dodaj_rezerwacje3(id_wycieczki_id
IN INT, id_osoby_id IN INT) AS
osoba  INT;
wycieczka  INT;
ile_wolnych INT;
ostatnia_rezerwacja INT;

BEGIN
 BEGIN
   SELECT   count(osoby.id_osoby)  into  osoba FROM  osoby
where OSOBY.ID_OSOBY = id_osoby_id;
   SELECT  count(wycieczki.id_wycieczki) into wycieczka from
WYCIECZKI where WYCIECZKI.ID_WYCIECZKI = id_wycieczki_id;
   SELECT  wycieczki.LICZBA_WOLNYCH_MIEJSC into ile_wolnych
from wycieczki where WYCIECZKI.ID_WYCIECZKI = id_wycieczki_id;
 END;
 if(osoba = 0 )
 THEN
   raise_application_error (-20001, 'Nie ma osoby o takim  id
' , TRUE);
 END IF ;
 if(wycieczka = 0 )
 THEN
   raise_application_error (-20001, 'Nie ma wycieczki o takim
id  ' , TRUE);
 END IF;
 if(ile_wolnych > 0)
 THEN
   INSERT INTO rezerwacje
   (id_wycieczki, id_osoby, status)
   VALUES( id_wycieczki_id, id_osoby_id, 'N');
```

```
   BEGIN
     SELECT max(REZERWACJE.NR_REZERWACJI) into
ostatnia_rezerwacja from REZERWACJE;
   END;
   INSERT  INTO REZERWACJE_LOG
   (NR_REZERWACJI,STATUS, DATA)
    values(ostatnia_rezerwacja, 'N',current_date ) ;
 end IF;
 RETURN;
END;
/
```

-aktualizacja liczby wolnych miejsc przy zmianie statusu rezerwacji

```
CREATE OR REPLACE TRIGGER zmien_status_rezerwacji
AFTER UPDATE OF STATUS ON REZERWACJE FOR EACH ROW
 DECLARE
   roznica INT := 0;
 BEGIN
   IF :OLD.STATUS != 'A' AND :NEW.STATUS = 'A' THEN
     roznica := 1;
   ELSEIF :OLD.STATUS = 'A' AND :NEW.STATUS != 'A' THEN
     roznica := -1;
   ELSE
     EXIT;
   END IF;
   UPDATE WYCIECZKI
   SET LICZBA_WOLNYCH_MIEJSC = LICZBA_WOLNYCH_MIEJSC - roznica
   WHERE ID_WYCIECZKI = :OLD.ID_WYCIECZKI;
 END;
```

```
CREATE PROCEDURE zmien_status_rezerwacji3(id_rezerwacji_id IN
INT, status IN char)
AS
rezerwacja INT;
stan char(1);
BEGIN
 BEGIN
   SELECT count(rezerwacje.nr_rezerwacji) into rezerwacja from
REZERWACJE where rezerwacje.NR_REZERWACJI = id_rezerwacji_id;
   SELECT rezerwacje.STATUS into stan from REZERWACJE where
REZERWACJE.NR_REZERWACJI = id_rezerwacji_id;
```

```
 end;
if (rezerwacja = 0)
 then
   raise_application_error (-20001, 'Nie ma wycieczki o takim
id ' , TRUE);
 end if;
 if stan = 'A'
 then
   raise_application_error (-20001, 'Wycieczka anulowana' ,
TRUE);
 end if;
 UPDATE rezerwacje
 SET status = 'P'
 WHERE nr_rezerwacji = id_rezerwacji_id;
return ;
END;
/
```

-aktualizacja liczby miejsc na poziomie wycieczki

```
create or replace TRIGGER AKTUALIZUJ_ZMIANA_LICZBY_MIEJSC
 BEFORE
   UPDATE OF LICZBA_MIEJSC
 ON WYCIECZKI
 FOR EACH ROW
 DECLARE
   ilosc number;
BEGIN

   SELECT count(*) INTO zmienna
       FROM REZERWACJE r WHERE r.ID_WYCIECZKI =
:NEW.ID_WYCIECZKI
           AND r.STATUS != 'A';
   :NEW.LICZBA_WOLNYCH_MIEJSC := (:new.LICZBA_MIEJSC - ilosc);
END;
```