3. Tworzenie widoków. Należy przygotować kilka widoków ułatwiających dostęp do danych. Należy zwrócić uwagę na strukturę kodu (należy unikać powielania kodu)

A. RzerwacjeWszystkie(kraj,data, nazwa_wycieczki, imie, nazwisko,status_rezerwacji)

```
CREATE VIEW RezerwacjeWszystkie
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;
```

B. RezerwacjePotwierdzone (kraj,data, nazwa_wycieczki, imie, nazwisko,status_rezerwacji)

```
CREATE VIEW RezerwacjePotwierdzone
AS
SELECT r.ID_WYCIECZKI,
      r.NAZWA,
      r.KRAJ,
      r.DATA,
      r.IMIE,
      r.NAZWISKO,
      r.STATUS
FROM REZERWACJEWSZYSTKIE R
   where STATUS = 'P'
```

C. RezerwacjeWPrzyszlosci (kraj,data, nazwa_wycieczki, imie, nazwisko,status_rezerwacji)

```
CREATE VIEW RezerwacjeWPrzyszlosci
AS
SELECT r.ID_WYCIECZKI,
      r.NAZWA,
      r.KRAJ,
      r.DATA,
      r.IMIE,
      r.NAZWISKO,
      r.STATUS
FROM REZERWACJEWSZYSTKIE R
   where r.DATA > CURRENT_DATE;
```

D. WycieczkiMiejsca(kraj,data, nazwa_wycieczki,liczba_miejsc, liczba_wolnych_miejsc)

```
CREATE VIEW WycieczkiMiejsca
AS
SELECT W.KRAJ,
      W.DATA,
      W.LICZBA_MIEJSC,
      W.LICZBA_MIEJSC - (select count(*) from REZERWACJE R where W.ID_WYCIECZKI = R.ID_WYCIECZKI) as
LICZBA_WOLNYCH_MIEJSC
FROM WYCIECZKI W
```

E. WycieczkiDostepne(kraj,data, nazwa_wycieczki,liczba_miejsc, liczba_wolnych_miejsc)

```
CREATE VIEW WycieczkiDostepne
AS
SELECT W.KRAJ,
      W.DATA,
      W.LICZBA_MIEJSC,
      W.LICZBA_WOLNYCH_MIEJSC
FROM WYCIECZKIMIEJSCA W
   where LICZBA_WOLNYCH_MIEJSC > 0
```

4.1. Tworzenie obiektów potrzebnych do działania funkcji

A. Wycieczka

```
create TYPE WYCIECZKA_OBJECT AS object
(
   id_wycieczki  NUMBER,
   nazwa         VARCHAR2(100),
   kraj          VARCHAR2(50),
   "data"        DATE,
   opis          VARCHAR2(200),
   liczba_miejsc NUMBER
)
create type WYCIECZKA_TYPE as table of WYCIECZKA_OBJECT
```

B. Osoba powiązana z wycieczką

```
create TYPE OSOBA_WYCIECZKA_OBJECT AS object
(
   kraj            VARCHAR(50),
   "data"          DATE,
   nazwa_wycieczki VARCHAR(100),
   imie            VARCHAR2(50),
   nazwisko        VARCHAR2(50),
   status          CHAR(1)
)

create type OSOBA_WYCIECZKA_TYPE as table of OSOBA_WYCIECZKA_OBJECT
```

4.2. Tworzenie procedur/funkcji pobierających dane. Podobnie jak w poprzednim przykładzie należy przygotować kilka procedur ułatwiających dostęp do danych:

A. UczestnicyWycieczki (id_wycieczki), procedura ma zwracać podobny zestaw danych jak widok RezerwacjeWszystkie

```
create FUNCTION UCZESTNICY_WYCIECZKI(id WYCIECZKI.ID_WYCIECZKI%TYPE)
   return OSOBA_WYCIECZKA_TYPE as
   tab_ret  OSOBA_WYCIECZKA_TYPE;
   flag integer ;
BEGIN
   SELECT COUNT(*)
   INTO flag
   FROM WYCIECZKI
   WHERE WYCIECZKI.ID_WYCIECZKI = id;

   IF flag = 0 THEN
       raise_application_error(-20001, '// WYCIECZKA O PODANYYM ID NIE ISTNIEJE // | An error was
encountered - ' || SQLCODE || ' -ERROR- ' || SQLERRM);
   END IF;

   SELECT OSOBA_WYCIECZKA_OBJECT(w.KRAJ, w.DATA, w.NAZWA, o.IMIE,o.NAZWISKO, r.STATUS) BULK COLLECT
   INTO tab_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 AND r.STATUS <> 'A';
   return tab_ret;
end UCZESTNICY_WYCIECZKI ;
```

B. RezerwacjeOsoby(id_osoby), procedura ma zwracać podobny zestaw danych jak widok wycieczki_osoby

```
create FUNCTION REZERWACJE_OSOBY(id OSOBY.ID_OSOBY%TYPE)
   return OSOBA_WYCIECZKA_TYPE as
   results  OSOBA_WYCIECZKA_TYPE;
   flag integer ;
BEGIN
   SELECT COUNT(*)
   INTO flag
   FROM OSOBY
   WHERE OSOBY.ID_OSOBY = id;

   IF flag = 0 THEN
       raise_application_error(-20001, '// OSOBA O PODANYYM ID NIE ISTNIEJE // | An error was
encountered - ' || SQLCODE || ' -ERROR- ' || SQLERRM);
   END IF;

   SELECT OSOBA_WYCIECZKA_OBJECT(w.KRAJ, w.DATA, w.NAZWA, o.IMIE,o.NAZWISKO, r.STATUS) BULK COLLECT
   INTO results
   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 AND r.STATUS <> 'A';
   return results;
end REZERWACJE_OSOBY ;
```

C. DostepneWycieczki(kraj, data_od, data_do)

```
create FUNCTION DOSTEPNE_WYCIECZKI(kraj WYCIECZKI.KRAJ%TYPE, data_od DATE, data_do DATE)
   return WYCIECZKA_TYPE as
   results  WYCIECZKA_TYPE;
BEGIN
   IF data_do < data_od
   THEN
       raise_application_error(-20001, '//DATA POCZATKOWA LUB KONCOWA ZLE USTAWIONE//' || SQLCODE ||
' -ERROR- ' || SQLERRM);
   END IF;

   SELECT WYCIECZKA_OBJECT(w.ID_WYCIECZKI, w.NAZWA, w.KRAJ, w.DATA, w.OPIS, w.LICZBA_MIEJSC) BULK
COLLECT
   INTO results
   FROM WYCIECZKI w
   WHERE w.KRAJ = DOSTEPNE_WYCIECZKI.kraj
     AND w.DATA >= data_od
     AND w.DATA <= data_do
     AND w.LICZBA_MIEJSC > (SELECT COUNT(*)
                            FROM REZERWACJE r
                            WHERE r.status <> 'A' AND r.ID_WYCIECZKI = w.ID_WYCIECZKI);
   return results;
end DOSTEPNE_WYCIECZKI ;
```

5. Tworzenie procedur modyfikujących dane. Należy przygotować zestaw procedur pozwalających na modyfikację danych oraz kontrolę poprawności ich wprowadzania

A. DodajRezerwacje(id_wycieczki, id_osoby), procedura powinna kontrolować czy wycieczka jeszcze się nie odbyła, i czy sa wolne miejsca

```
create PROCEDURE DODAJ_REZERWACJE(id_wycieczki WYCIECZKI.ID_WYCIECZKI%TYPE, id_osoby
OSOBY.ID_OSOBY%TYPE) AS
   id_dodanej_rezerwacji integer;
   flag                  integer;
BEGIN
   SELECT COUNT(*)
   INTO flag
   FROM OSOBY
   WHERE OSOBY.ID_OSOBY = DODAJ_REZERWACJE.id_osoby;

   IF flag = 0 THEN
       raise_application_error(-20001, '//NIE MA OSOBY Z TAKIM ID// ' || SQLCODE || ' -ERROR- ' ||
SQLERRM);
```

```
   END IF;

   SELECT COUNT(*)
   INTO flag
   FROM WYCIECZKIDOSTEPNE w
   WHERE w.ID_WYCIECZKI = DODAJ_REZERWACJE.id_wycieczki;

   IF flag = 0 THEN
       raise_application_error(-20001, '//PODANA WYCIECZKA NIE JEST DOSTEOPNA/NIE ISTNIEJE// ' ||
SQLCODE || ' -ERROR- ' || SQLERRM);
   END IF;

   SELECT COUNT(*)
   INTO flag
   FROM REZERWACJE r
   WHERE r.ID_WYCIECZKI = DODAJ_REZERWACJE.id_wycieczki AND r.ID_OSOBY = DODAJ_REZERWACJE.id_osoby;

   IF flag > 0 THEN
       raise_application_error(-20001, '//REZERWACJA ZOSTALA JUZ ZROBIONA// ' || SQLCODE || '
-ERROR- ' || SQLERRM);
   END IF;

   INSERT INTO REZERWACJE(id_wycieczki, id_osoby, STATUS)
   VALUES(DODAJ_REZERWACJE.id_wycieczki, DODAJ_REZERWACJE.id_osoby, 'N') returning NR_REZERWACJI
into id_dodanej_rezerwacji ;

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

END DODAJ_REZERWACJE;
```

B. ZmienStatusRezerwacji(nr_rezerwacji, status), procedura kontrolować czy możliwa jest zmiana statusu, np. zmiana statusu już anulowanej wycieczki (przywrócenie do stanu aktywnego nie zawsze jest możliwa – może już nie być miejsc)

```
create PROCEDURE ZMIEN_STATUS_REZERWACJI(nr_rezerwacji REZERWACJE.ID_WYCIECZKI%TYPE,status
REZERWACJE.STATUS%TYPE) AS
   flag         integer;
   stary_status REZERWACJE.STATUS%TYPE;

BEGIN
   SELECT count(*)
   INTO flag
   FROM WYCIECZKIDOSTEPNE wp
            JOIN REZERWACJE r ON r.ID_WYCIECZKI = wp.ID_WYCIECZKI
   WHERE r.NR_REZERWACJI = ZMIEN_STATUS_REZERWACJI.nr_rezerwacji;

   IF flag = 0 THEN
       raise_application_error(-20001, '//WYCIECZKA JUZ SIE ODBYLA //' || SQLCODE || ' -ERROR- ' ||
SQLERRM);
   end IF;

   SELECT status
   INTO stary_status
   FROM REZERWACJE
   WHERE NR_REZERWACJI = ZMIEN_STATUS_REZERWACJI.nr_rezerwacji;

   CASE
```

```
       WHEN stary_status IS NULL THEN
           raise_application_error(-20001, '//REZERWACJA NIE ISTNIEJE!//' || SQLCODE || ' -ERROR- '
|| SQLERRM);

       WHEN ZMIEN_STATUS_REZERWACJI.status = 'N' THEN
           raise_application_error(-20001, '// NIE MOZNA ISTANIEJACEJ REZERWACJI ZROBIC JAKO
"NOWA"//' || SQLCODE || ' -ERROR- ' || SQLERRM);

       WHEN stary_status = 'A' THEN
           SELECT COUNT(*)
           INTO flag
           FROM WYCIECZKIDOSTEPNE wd
                    JOIN REZERWACJE r ON r.ID_WYCIECZKI = wd.ID_WYCIECZKI
           WHERE r.NR_REZERWACJI = ZMIEN_STATUS_REZERWACJI.nr_rezerwacji;
           IF flag = 0 THEN
              raise_application_error(-20001, '// NIE MOZNA ISTANIEJACEJ REZERWACJI ZROBIC JAKO
"NOWA"//' || SQLCODE || ' -ERROR- ' || SQLERRM);
           END IF;
   END CASE;

   UPDATE REZERWACJE
   SET STATUS = ZMIEN_STATUS_REZERWACJI.status
   WHERE NR_REZERWACJI = ZMIEN_STATUS_REZERWACJI.nr_rezerwacji;

   INSERT INTO REZERWACJE_LOG(ID_REZERWACJI, DATA, STATUS) VALUES (nr_rezerwacji, CURRENT_DATE,
status);
END ZMIEN_STATUS_REZERWACJI;
```

C. ZmienLiczbeMiejsc(id_wycieczki, liczba_miejsc), nie wszystkie zmiany liczby miejsc są dozwolone, nie można zmniejszyć liczby miesc na wartość poniżej liczby zarezerwowanych miejsc

```
create PROCEDURE ZMIEN_LICZBE_MIEJSC(id_wycieczki WYCIECZKI.ID_WYCIECZKI%TYPE, liczba_miejsc
WYCIECZKI.LICZBA_MIEJSC%TYPE) AS
   zajete_miejsca integer ;
BEGIN
   SELECT wm.LICZBA_MIEJSC - wm.LICZBA_WOLNYCH_MIEJSC
   INTO zajete_miejsca
   FROM WYCIECZKIMIEJSCA wm
   WHERE wm.ID_WYCIECZKI = ZMIEN_LICZBE_MIEJSC.id_wycieczki;

   IF ZMIEN_LICZBE_MIEJSC.liczba_miejsc < 0 OR zajete_miejsca > ZMIEN_LICZBE_MIEJSC.liczba_miejsc
THEN
       raise_application_error(-20001, '//OBIZKA O ZBYT DUZA LICZBE MIEJSC//' || SQLCODE || '
-ERROR- ' || SQLERRM);
   end if;

   UPDATE WYCIECZKI
   SET LICZBA_MIEJSC = ZMIEN_LICZBE_MIEJSC.liczba_miejsc
   WHERE ID_WYCIECZKI = ZMIEN_LICZBE_MIEJSC.id_wycieczki;
END ZMIEN_LICZBE_MIEJSC;
```

6. Dodajemy tabelę dziennikującą zmiany statusu rezerwacji

```
create table REZERWACJE_LOG
(
   ID            NUMBER generated as identity
       constraint REZERWACJE_LOG_PK
           primary key,
   ID_REZERWACJI NUMBER,
   DATA          DATE,
   STATUS        CHAR
)
```

7. Zmiana struktury bazy danych, w tabeli wycieczki dodajemy redundantne pole liczba_wolnych_miejsc

A. Dodanie liczby wolnych miejsc do tabeli wycieczki

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

B. Zmiana procedury dodaj rezerwacje

```
create PROCEDURE DODAJ_REZERWACJE2(id_wycieczki WYCIECZKI.ID_WYCIECZKI%TYPE, id_osoby
OSOBY.ID_OSOBY%TYPE) AS
   id_dodanej_rezerwacji integer;
   flag                  integer;
BEGIN
   SELECT COUNT(*)
   INTO flag
   FROM OSOBY
   WHERE OSOBY.ID_OSOBY = DODAJ_REZERWACJE2.id_osoby;

   IF flag = 0 THEN
       raise_application_error(-20001, '//NIE MA OSOBY Z TAKIM ID// ' || SQLCODE || ' -ERROR- ' ||
SQLERRM);
   END IF;

   SELECT COUNT(*)
   INTO flag
   FROM WYCIECZKIDOSTEPNE w
   WHERE w.ID_WYCIECZKI = DODAJ_REZERWACJE2.id_wycieczki;

   IF flag = 0 THEN
       raise_application_error(-20001, '//PODANA WYCIECZKA NIE JEST DOSTEOPNA/NIE ISTNIEJE// ' ||
SQLCODE || ' -ERROR- ' || SQLERRM);
   END IF;
```

```
   SELECT COUNT(*)
   INTO flag
   FROM REZERWACJE r
   WHERE r.ID_WYCIECZKI = DODAJ_REZERWACJE2.id_wycieczki AND r.ID_OSOBY =
DODAJ_REZERWACJE2.id_osoby;

   IF flag > 0 THEN
       raise_application_error(-20001, '//REZERWACJA ZOSTALA JUZ ZROBIONA// ' || SQLCODE || '
-ERROR- ' || SQLERRM);
   END IF;

   INSERT INTO REZERWACJE(id_wycieczki, id_osoby, STATUS)
   VALUES(DODAJ_REZERWACJE2.id_wycieczki, DODAJ_REZERWACJE2.id_osoby, 'N') returning NR_REZERWACJI
into id_dodanej_rezerwacji ;

   UPDATE WYCIECZKI
   SET LICZBA_WOLNYCH_MIEJSC = LICZBA_WOLNYCH_MIEJSC - 1
   WHERE ID_WYCIECZKI = DODAJ_REZERWACJE2.id_wycieczki;

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

END DODAJ_REZERWACJE2;
```

C. Zmiana procedury ZMIEN_STATUS_REZERWACJI

```
create PROCEDURE ZMIEN_STATUS_REZERWACJI2(nr_rezerwacji REZERWACJE.ID_WYCIECZKI%TYPE,status
REZERWACJE.STATUS%TYPE) AS
   flag         integer;
   stary_status REZERWACJE.STATUS%TYPE;

BEGIN
   SELECT count(*)
   INTO flag
   FROM WYCIECZKIDOSTEPNE wp
            JOIN REZERWACJE r ON r.ID_WYCIECZKI = wp.ID_WYCIECZKI
   WHERE r.NR_REZERWACJI = ZMIEN_STATUS_REZERWACJI2.nr_rezerwacji;

   IF flag = 0 THEN
       raise_application_error(-20001, '//WYCIECZKA JUZ SIE ODBYLA //' || SQLCODE || ' -ERROR- ' ||
SQLERRM);
   end IF;

   SELECT status
   INTO stary_status
   FROM REZERWACJE
   WHERE NR_REZERWACJI = ZMIEN_STATUS_REZERWACJI2.nr_rezerwacji;

   CASE
       WHEN stary_status IS NULL THEN
           raise_application_error(-20001, '//REZERWACJA NIE ISTNIEJE!//' || SQLCODE || ' -ERROR- '
|| SQLERRM);

       WHEN ZMIEN_STATUS_REZERWACJI2.status = 'N' THEN
           raise_application_error(-20001, '// NIE MOZNA ISTANIEJACEJ REZERWACJI ZROBIC JAKO
"NOWA"//' || SQLCODE || ' -ERROR- ' || SQLERRM);

       WHEN stary_status = 'A' THEN
           SELECT COUNT(*)
```

```
            INTO flag
            FROM WYCIECZKIDOSTEPNE wd
                     JOIN REZERWACJE r ON r.ID_WYCIECZKI = wd.ID_WYCIECZKI
            WHERE r.NR_REZERWACJI = ZMIEN_STATUS_REZERWACJI2.nr_rezerwacji;
            IF flag = 0 THEN
               raise_application_error(-20001, '// NIE MOZNA ISTANIEJACEJ REZERWACJI ZROBIC JAKO
"NOWA"//' || SQLCODE || ' -ERROR- ' || SQLERRM);
            END IF;
   END CASE;

   UPDATE REZERWACJE
   SET STATUS = ZMIEN_STATUS_REZERWACJI2.status
   WHERE NR_REZERWACJI = ZMIEN_STATUS_REZERWACJI2.nr_rezerwacji;

   UPDATE WYCIECZKI w
   SET LICZBA_WOLNYCH_MIEJSC = LICZBA_WOLNYCH_MIEJSC - 1
   WHERE w.ID_WYCIECZKI = (SELECT ID_WYCIECZKI
                           FROM REZERWACJE r
                           WHERE r.NR_REZERWACJI = ZMIEN_STATUS_REZERWACJI2.nr_rezerwacji);

   INSERT INTO REZERWACJE_LOG(ID_REZERWACJI, DATA, STATUS) VALUES (nr_rezerwacji, CURRENT_DATE,
status);
END ZMIEN_STATUS_REZERWACJI2;
```

D. Zmiana procedury `ZMIEN_LICZBE_MIEJSC`

```
create PROCEDURE ZMIEN_LICZBE_MIEJSC2(id_wycieczki WYCIECZKI.ID_WYCIECZKI%TYPE, liczba_miejsc
WYCIECZKI.LICZBA_MIEJSC%TYPE) AS
   zajete_miejsca integer ;
BEGIN
   SELECT wm.LICZBA_MIEJSC - wm.LICZBA_WOLNYCH_MIEJSC
   INTO zajete_miejsca
   FROM WYCIECZKIMIEJSCA wm
   WHERE wm.ID_WYCIECZKI = ZMIEN_LICZBE_MIEJSC2.id_wycieczki;

   IF ZMIEN_LICZBE_MIEJSC2.liczba_miejsc < 0 OR zajete_miejsca > ZMIEN_LICZBE_MIEJSC2.liczba_miejsc
THEN
       raise_application_error(-20001, '//OBIZKA O ZBYT DUZA LICZBE MIEJSC//' || SQLCODE || '
-ERROR- ' || SQLERRM);
   end if;

   UPDATE WYCIECZKI
   SET LICZBA_MIEJSC = ZMIEN_LICZBE_MIEJSC2.liczba_miejsc,
       LICZBA_WOLNYCH_MIEJSC = LICZBA_WOLNYCH_MIEJSC + (ZMIEN_LICZBE_MIEJSC2.liczba_miejsc -
LICZBA_MIEJSC)
   WHERE ID_WYCIECZKI = ZMIEN_LICZBE_MIEJSC2.id_wycieczki;
END ZMIEN_LICZBE_MIEJSC2;
```

E. zmiana funkcji `DOSTEPNE_WYCIECZKI2`

```
create FUNCTION DOSTEPNE_WYCIECZKI2(kraj WYCIECZKI.KRAJ%TYPE, data_od DATE, data_do DATE)
   return WYCIECZKA_TYPE as
   results  WYCIECZKA_TYPE;
BEGIN
   IF data_do < data_od
   THEN
       raise_application_error(-20001, '//DATA POCZATKOWA LUB KONCOWA ZLE USTAWIONE//' || SQLCODE ||
' -ERROR- ' || SQLERRM);
   END IF;
```

```
   SELECT WYCIECZKA_OBJECT(w.ID_WYCIECZKI, w.NAZWA, w.KRAJ, w.DATA, w.OPIS, w.LICZBA_MIEJSC) BULK
COLLECT
   INTO results
   FROM WYCIECZKI w
   WHERE w.KRAJ = DOSTEPNE_WYCIECZKI2.kraj
     AND w.DATA >= data_od
     AND w.DATA <= data_do
     AND w.LICZBA_WOLNYCH_MIEJSC > 0;
   return results;
end DOSTEPNE_WYCIECZKI2 ;
```

8. Zmiana strategii zapisywania do dziennika rezerwacji. Realizacja przy pomocy triggerów.

A. Trigger do dodania rezerwacji

```
CREATE OR REPLACE TRIGGER DODANIE_REZERWACJI_TRIGGER
   AFTER UPDATE
   ON REZERWACJE
   FOR EACH ROW
DECLARE
   counter 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
           counter := -1;
       WHEN
           :OLD.STATUS <> 'A' AND :NEW.STATUS = 'A' THEN
           counter := 1;
       ELSE
           counter := 0;
   END CASE;

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

B. Trigger do zmiany statusu

```
CREATE OR REPLACE TRIGGER ZMIANA_STATUSU_TRIGGER
   AFTER UPDATE
   ON REZERWACJE
   FOR EACH ROW
DECLARE
   counter 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
           counter := -1;
       WHEN :OLD.STATUS <> 'A' AND :NEW.STATUS = 'A' THEN
           counter := 1;
       ELSE
           counter := 0;
   END CASE;

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

C. Trigger zabraniający usuwać rezerwacje

```
CREATE OR REPLACE TRIGGER USUNIECIE_REZERWACJI_TRIGGER
   BEFORE DELETE
   ON REZERWACJE
   FOR EACH ROW
BEGIN
   raise_application_error(-20001, '//NIE MOZNA USUNAC REZERWACJI!//' || SQLCODE || ' -ERROR- ' ||
SQLERRM);
end;
```

D. zmiana procedury `DODAJ_REZERWACJE` aby dziala z triggerami

```
create PROCEDURE DODAJ_REZERWACJE3(id_wycieczki WYCIECZKI.ID_WYCIECZKI%TYPE, id_osoby
OSOBY.ID_OSOBY%TYPE) AS
   id_dodanej_rezerwacji integer;
   flag                  integer;
BEGIN
   SELECT COUNT(*)
   INTO flag
   FROM OSOBY
   WHERE OSOBY.ID_OSOBY = DODAJ_REZERWACJE3.id_osoby;

   IF flag = 0 THEN
       raise_application_error(-20001, '//NIE MA OSOBY Z TAKIM ID// ' || SQLCODE || ' -ERROR- ' ||
SQLERRM);
   END IF;

   SELECT COUNT(*)
   INTO flag
   FROM WYCIECZKIDOSTEPNE w
   WHERE w.ID_WYCIECZKI = DODAJ_REZERWACJE3.id_wycieczki;

   IF flag = 0 THEN
       raise_application_error(-20001, '//PODANA WYCIECZKA NIE JEST DOSTEOPNA/NIE ISTNIEJE// ' ||
SQLCODE || ' -ERROR- ' || SQLERRM);
   END IF;

   SELECT COUNT(*)
   INTO flag
   FROM REZERWACJE r
   WHERE r.ID_WYCIECZKI = DODAJ_REZERWACJE3.id_wycieczki AND r.ID_OSOBY =
DODAJ_REZERWACJE3.id_osoby;
```

```
   IF flag > 0 THEN
       raise_application_error(-20001, '//REZERWACJA ZOSTALA JUZ ZROBIONA// ' || SQLCODE || '
-ERROR- ' || SQLERRM);
   END IF;

   INSERT INTO REZERWACJE(id_wycieczki, id_osoby, STATUS)
   VALUES(DODAJ_REZERWACJE3.id_wycieczki, DODAJ_REZERWACJE3.id_osoby, 'N') returning NR_REZERWACJI
into id_dodanej_rezerwacji ;
END DODAJ_REZERWACJE3;
```

E. zmiana procedury `ZMIEN_STATUS_REZERWACJI` aby dziala z triggerami

```
create PROCEDURE ZMIEN_STATUS_REZERWACJI3(nr_rezerwacji REZERWACJE.ID_WYCIECZKI%TYPE,status
REZERWACJE.STATUS%TYPE) AS
   flag         integer;
   stary_status REZERWACJE.STATUS%TYPE;

BEGIN
   SELECT count(*)
   INTO flag
   FROM WYCIECZKIDOSTEPNE wp
            JOIN REZERWACJE r ON r.ID_WYCIECZKI = wp.ID_WYCIECZKI
   WHERE r.NR_REZERWACJI = ZMIEN_STATUS_REZERWACJI3.nr_rezerwacji;

   IF flag = 0 THEN
       raise_application_error(-20001, '//WYCIECZKA JUZ SIE ODBYLA //' || SQLCODE || ' -ERROR- ' ||
SQLERRM);
   end IF;

   SELECT status
   INTO stary_status
   FROM REZERWACJE
   WHERE NR_REZERWACJI = ZMIEN_STATUS_REZERWACJI3.nr_rezerwacji;

   CASE
       WHEN stary_status IS NULL THEN
           raise_application_error(-20001, '//REZERWACJA NIE ISTNIEJE!//' || SQLCODE || ' -ERROR- '
|| SQLERRM);

       WHEN ZMIEN_STATUS_REZERWACJI3.status = 'N' THEN
           raise_application_error(-20001, '// NIE MOZNA ISTANIEJACEJ REZERWACJI ZROBIC JAKO
"NOWA"//' || SQLCODE || ' -ERROR- ' || SQLERRM);

       WHEN stary_status = 'A' THEN
           SELECT COUNT(*)
           INTO flag
           FROM WYCIECZKIDOSTEPNE wd
                    JOIN REZERWACJE r ON r.ID_WYCIECZKI = wd.ID_WYCIECZKI
           WHERE r.NR_REZERWACJI = ZMIEN_STATUS_REZERWACJI3.nr_rezerwacji;
           IF flag = 0 THEN
              raise_application_error(-20001, '// NIE MOZNA ISTANIEJACEJ REZERWACJI ZROBIC JAKO
"NOWA"//' || SQLCODE || ' -ERROR- ' || SQLERRM);
           END IF;
   END CASE;

   UPDATE REZERWACJE
   SET STATUS = ZMIEN_STATUS_REZERWACJI3.status
   WHERE NR_REZERWACJI = ZMIEN_STATUS_REZERWACJI3.nr_rezerwacji;
END ZMIEN_STATUS_REZERWACJI3;
```

F. zmiana procedury `ZMIEN_LICZBE_MIEJSC` aby działała z triggerami

```
create PROCEDURE ZMIEN_LICZBE_MIEJSC3(id_wycieczki WYCIECZKI.ID_WYCIECZKI%TYPE, liczba_miejsc
WYCIECZKI.LICZBA_MIEJSC%TYPE) AS
   zajete_miejsca integer ;
BEGIN
   SELECT wm.LICZBA_MIEJSC - wm.LICZBA_WOLNYCH_MIEJSC
   INTO zajete_miejsca
   FROM WYCIECZKIMIEJSCA wm
   WHERE wm.ID_WYCIECZKI = ZMIEN_LICZBE_MIEJSC3.id_wycieczki;

   IF ZMIEN_LICZBE_MIEJSC3.liczba_miejsc < 0 OR zajete_miejsca > ZMIEN_LICZBE_MIEJSC3.liczba_miejsc
THEN
       raise_application_error(-20001, '//OBIZKA O ZBYT DUZA LICZBE MIEJSC//' || SQLCODE || '
-ERROR- ' || SQLERRM);
   end if;

   UPDATE WYCIECZKI
   SET LICZBA_MIEJSC = ZMIEN_LICZBE_MIEJSC3.liczba_miejsc
   WHERE ID_WYCIECZKI = ZMIEN_LICZBE_MIEJSC3.id_wycieczki;
END ZMIEN_LICZBE_MIEJSC3;
```