1.Tabele

Pole status w tabeli Rezerwacje może przyjmować jedną z 4 wartości

N – Nowa

P – Potwierdzona

Z – Potwierdzona i zapłacona

A – Anulowana

```
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
);
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
);
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
);
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.Wypełnianie tabele przykładowymi danymi danymi

4 wycieczki
10 osób
10 rezerwacji

Dane testowe powinny być różnorodne (wycieczki w przyszłości, wycieczki w przeszłości, rezerwacje o różnym statusie itp.) tak, żeby umożliwić testowanie napisanych procedur. W razie potrzeby należy zmodyfikować dane tak żeby przetestować różne przypadki.

```
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, dzwonić po 18.00');

INSERT INTO osoby (imie, nazwisko, pesel, kontakt)
VALUES('Stanuisław', 'Hopin', '87984765', 'tel: 4567');

INSERT INTO osoby (imie, nazwisko, pesel, kontakt)
VALUES('Konrad', 'Dębiec', '18890543', 'tel: 3458, nie dzwonić');

INSERT INTO osoby (imie, nazwisko, pesel, kontakt)
VALUES('Kasia', 'Louren', '56789438', 'tel: 7894');

INSERT INTO osoby (imie, nazwisko, pesel, kontakt)
VALUES('Sem', 'Jonowski', '97654321', 'tel: 6789');

INSERT INTO osoby (imie, nazwisko, pesel, kontakt)
VALUES('Ema', 'Indor', '91234567', 'tel: 1233');

INSERT INTO osoby (imie, nazwisko, pesel, kontakt)
VALUES('Lauren', 'Filaja', '12873465', 'tel: 2444, dzwonić przed 20.00');

INSERT INTO osoby (imie, nazwisko, pesel, kontakt)
VALUES('Tom', 'Paule', '12348765', 'tel: 6677');

INSERT INTO osoby (imie, nazwisko, pesel, kontakt)
VALUES('Jon', 'West', '43215678', 'tel: 7899, dzwonić po 14.00');
-----------------------------------------------------------------------------------------------
INSERT INTO wycieczki (nazwa, kraj, data, opis, liczba_miejsc)
VALUES ('Wycieczka do Paryza','Francja',TO_DATE('2016-04-03','YYYY-MM-DD'),'Ciekawa wycieczka
...',3);
```

```
INSERT INTO wycieczki (nazwa, kraj, data, opis, liczba_miejsc)
VALUES ('Piękny Kraków','Polska',TO_DATE('2017-05-06','YYYY-MM-DD'),'Najciekawa wycieczka ...',2);

INSERT INTO wycieczki (nazwa, kraj, data, opis, liczba_miejsc)
VALUES ('Wieliczka','Polska',TO_DATE('2021-07-09','YYYY-MM-DD'),'Zadziwiająca kopalnia ...',2);

INSERT INTO wycieczki (nazwa, kraj, data, opis, liczba_miejsc)
VALUES ('Tajemnicy Lublina','Polska',TO_DATE('2020-04-05','YYYY-MM-DD'),'Super wrażenia ...',4);
```

```
INSERT INTO rezerwacje(id_wycieczki, id_osoby, status)
VALUES (2,1,'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,'A');
INSERT INTO rezerwacje(id_wycieczki, id_osoby, status)
VALUES (5,5,'N');
INSERT INTO rezerwacje(id_wycieczki, id_osoby, status)
VALUES (3,6,'P');
INSERT INTO rezerwacje(id_wycieczki, id_osoby, status)
VALUES (2,7,'Z');
INSERT INTO rezerwacje(id_wycieczki, id_osoby, status)
VALUES (3,8,'A');
INSERT INTO rezerwacje(id_wycieczki, id_osoby, status)
VALUES (4,9,'N');
INSERT INTO rezerwacje(id_wycieczki, id_osoby, status)
VALUES (5,10,'P');
```

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

a) RezerwacjeWszystkie
(kraj,data, nazwa_wycieczki, imie, nazwisko,status_rezerwacji)

b) RezerwacjePotwierdzone
(kraj,data, nazwa_wycieczki, imie, nazwisko,status_rezerwacji)

c) RezerwacjeWPrzyszlosci
(kraj,data, nazwa_wycieczki, imie, nazwisko,status_rezerwacji)

d) WycieczkiMiejsca
(kraj,data, nazwa_wycieczki,liczba_miejsc, liczba_wolnych_miejsc)

e) WycieczkiDostepne
(kraj,data, nazwa_wycieczki,liczba_miejsc, liczba_wolnych_miejsc)

```
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
```

```sql
create view rezerwacjepotwierdzone
as
select *
from RezerwacjeWszystkie w
where w.status='P';
-----------------------------------------------------------------------------------------------------
create view RezerwacjeWPrzyslosci
as select *
from RezerwacjeWszystkie
where data>current_date;
-----------------------------------------------------------------------------------------------------
create view WycieczkiMiejsca
as select
w.
w.kraj,
w.data,
w.nazwa,
w.LICZBA_MIEJSC,
(w.LICZBA_MIEJSC-(select count(*) from rezerwacje r where r.ID_WYCIECZKI=w.id_wycieczki)) as
liczba_wolnych_miejsc
from wycieczki w;
-----------------------------------------------------------------------------------------------------
create view wycieczkidostepne
as select * from WycieczkiMiejsca
where LICZBA_MIEJSC>licliczba_wolnych_miejsc
```

4. 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
b) RezerwacjeOsoby(id_osoby),
procedura ma zwracać podobny zestaw danych jak widok wycieczki_osoby
c) DostepneWycieczki(kraj, data_od, data_do)

Procedury/funkcje powinny zwracać tabelę/zbiór wynikowy Należy zwrócić uwagę na kontrolę parametrów (np. jeśli parametrem jest id_wycieczki to należy sprawdzić czy taka wycieczka istnieje). Podobnie jak w przypadku widoków należy unikać powielania kodu.

```sql
CREATE OR REPLACE TYPE osoby_wycieczki_r AS object
(
   kraj            VARCHAR(50),
   "data"          DATE,
   nazwa_wycieczki VARCHAR(100),
   imie            VARCHAR2(50),
   nazwisko        VARCHAR2(50),
   status          CHAR(1)
);


CREATE OR REPLACE TYPE osoby_wycieczki_t IS TABLE OF osoby_wycieczki_r;
```

```
CREATE OR REPLACE
   FUNCTION uczestnicy_wycieczki(id INT)
   return osoby_wycieczki_t as
   w_retar  osoby_wycieczki_t;
   exist integer;
BEGIN
   SELECT COUNT(*) INTO exist FROM WYCIECZKI WHERE WYCIECZKI.ID_WYCIECZKI = id;

   IF exist = 0 THEN
       raise_application_error(-25000, 'Such trip does not exist');
   END IF;

   SELECT osoby_wycieczki_r(w.KRAJ, w.DATA, w.NAZWA, o.IMIE,
                            o.NAZWISKO, r.STATUS)
       BULK COLLECT
   INTO w_retar
   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 w_retar;
end uczestnicy_wycieczki;

-----------------------------------------------------------------------------------------------------
create or replace TYPE RezerwacjeOsobyO as Object
(
   ID_WYCIECZKI int,
   NAZWA        varchar2(100),
   KRAJ         varchar2(50),
   "DATA"       date,
   IMIE         varchar2(50),
   NAZWISKO     varchar2(50),
   STATUS       char(1)
);

create or replace TYPE TableRezerwacjeOsoby as TABLE of RezerwacjeOsobyO;

create or replace FUNCTION RezerwacjeOsoby(id_person INT)
   RETURN TableRezerwacjeOsoby as
   tableResult TableRezerwacjeOsoby;
   findPerson  integer;
BEGIN
   SELECT COUNT(*)
   INTO findPerson
   FROM OSOBY o
   WHERE o.ID_OSOBY = id_person;
   IF findPerson = 0 THEN
       RAISE_APPLICATION_ERROR(-30007, 'This person does not exist');
   END IF;

   SELECT RezerwacjeOsobyO(w.ID_WYCIECZKI, w.NAZWA, w.KRAJ, w.DATA, o.IMIE, o.NAZWISKO, r.STATUS)
       BULK COLLECT
   INTO tableResult
   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_person;
   RETURN tableResult;
```

```
END RezerwacjeOsoby;
--------------------------------------------------------------------------------------------------
create or replace TYPE ObjektDostepneWycieczki as Object
(
   ID_Wycieczki   int,
   NAZWA          varchar2(100),
   KRAJ           varchar2(50),
   "DATA"         date,
   LICZBA_MIEJSC  int,
   LICZBA_WOLNYCH int
);

create or replace TYPE TableDostepneWycieczki as TABLE of ObjektDostepneWycieczki;

create or replace FUNCTION DostepneWycieczki(country WYCIECZKI.KRAJ%TYPE, date_od date, date_do date)
   RETURN TableDostepneWycieczki as
   tableResult TableDostepneWycieczki;
BEGIN
   SELECT ObjektDostepneWycieczki(NAZWA, KRAJ, DATA, LICZBA_MIEJSC, LICZBA_WOLNYCH_MIEJSC)
       BULK COLLECT
   INTO tableResult
   FROM WycieczkiDostepne
   where KRAJ = country
     and DATA >= date_od
     and DATA <= date_do
     and LICZBA_WOLNYCH_MIEJSC > 0;
   RETURN tableResult;
END DostepneWycieczki;
```

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

Należy rozważyć użycie transakcji Należy zwrócić uwagę na kontrolę parametrów (np. jeśli parametrem jest id_wycieczki to należy sprawdzić czy taka wycieczka istnieje, jeśli robimy rezerwację to należy sprawdzać czy są wolne miejsca itp..)

```
CREATE OR REPLACE PROCEDURE
   dodaj_rezerwacje(id_wycieczki int,
                    id_osoby int) AS
   exist  integer;
   new_id integer;
BEGIN
   SELECT COUNT(*)
   INTO exist
   FROM OSOBY
   WHERE OSOBY.ID_OSOBY = dodaj_rezerwacje.id_osoby;
```

```
   IF exist = 0
   THEN
       raise_application_error(-30009, 'The person with given id does not exist');
   END IF;

   SELECT COUNT(*)
   INTO exist
   FROM WYCIECZKIDOSTEPNE w
   WHERE w.ID_WYCIECZKI = dodaj_rezerwacje.id_wycieczki;

   IF exist = 0
   THEN
       raise_application_error(-30008, 'The trip with given id does not exist');
   END IF;

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

   IF exist > 0
   THEN
       raise_application_error(-30007, 'The reservation already exist');
   END IF;

   INSERT INTO REZERWACJE (id_wycieczki, id_osoby, STATUS)
   VALUES (dodaj_rezerwacje.id_wycieczki, dodaj_rezerwacje.id_osoby, 'N');
END dodaj_rezerwacje;
-----------------------------------------------------------------------------------------------
create or replace Procedure ZmienStatusRezerwacji2(id_res REZERWACJE.NR_REZERWACJI%TYPE, stat
REZERWACJE.STATUS%TYPE) as
  canRes    integer;
  id_tour   int;
  count_res int;
calculateFreePlace int;

BEGIN
  SELECT COUNT(*)
  INTO count_res
  FROM REZERWACJE
  WHERE REZERWACJE.NR_REZERWACJI = id_res;
  IF count_res = 0 THEN
      RAISE_APPLICATION_ERROR(-30067, 'Given reservation does not exist');
  END IF;
IF ZmienStatusRezerwacji2.stat = 'A' THEN
          select ID_WYCIECZKI INTO id_tour from REZERWACJE r
          where r.NR_REZERWACJI = id_res;

          select count(*) INTO canRes from WycieczkiDostepne wd
          where wd.ID_WYCIECZKI =  id_tour;
          IF canRes = 0 THEN
              RAISE_APPLICATION_ERROR(-20011, 'There are no free places');
          ELSE
            calculateFreePlace := -1;
          END IF;
      ELSE
        IF stat = 'A'
```

```
        THEN
          calculateFreePlace := +1;
        ELSE
          calculateFreePlace := 0;
        end if;
      END If;

      UPDATE REZERWACJE
      SET STATUS = stat
      WHERE NR_REZERWACJI = id_res;

      INSERT INTO REZERWACJE_LOG(ID_REZERWACJI, DATA, STATUS)
      VALUES (id_res, CURRENT_DATE, stat);

      UPDATE WYCIECZKI w
      SET LICZBA_WOLNYCH_MIEJSC = LICZBA_WOLNYCH_MIEJSC + calculateFreePlace
      WHERE w.ID_WYCIECZKI = id_tour;
  end;

-------------------------------------------------------------------------------------------------
CREATE OR REPLACE PROCEDURE zmien_liczbe_miejsc(id_w WYCIECZKI.ID_WYCIECZKI%TYPE,
                                                nowe_miejsca WYCIECZKI.LICZBA_MIEJSC%TYPE) AS
   occupied integer;
   exist    integer;
BEGIN
   SELECT COUNT(*) INTO exist FROM WYCIECZKI WHERE WYCIECZKI.ID_WYCIECZKI = id_w;

   IF exist = 0 THEN
       raise_application_error(-25000, 'Dana wycieczka nie exist');
   END IF;

   BEGIN
       SELECT LICZBA_MIEJSC - LICZBA_WOLNYCH_MIEJSC
       INTO occupied
       FROM WYCIECZKIMIEJSCA wm
       WHERE wm.ID_WYCIECZKI = id_w;

       IF nowe_miejsca < 0 OR occupied > nowe_miejsca
       THEN
           raise_application_error(-20006, 'New places number is too low');
       end if;

       UPDATE WYCIECZKI
       SET LICZBA_MIEJSC = nowe_miejsca
       WHERE ID_WYCIECZKI = id_w;
   END;
End;
```

6. Dodajemy tabelę dziennikującą zmiany statusu rezerwacji Rezerwacje_log(id, id_rezerwacji, data, status)

Rezerwacje_log(id, id_rezerwacji, data, status)

Należy zmienić warstwę procedur modyfikujących dane tak aby dopisywały informację do dziennika

```
CREATE TABLE REZERWACJE_LOG
(
```

```
ID_REZERWACJE_LOG INT GENERATED ALWAYS AS IDENTITY NOT NULL,
ID_REZERWACJI INT,
"DATA" DATE,
STATUS CHAR(1),
CONSTRAINT REZERWACJE_LOG_PK PRIMARY KEY
(ID_REZERWACJE_LOG)
ENABLE
);

ALTER TABLE REZERWACJE_LOG
ADD CONSTRAINT REZERWACJE_LOG_FK1 FOREIGN KEY
(ID_REZERWACJI)
REFERENCES REZERWACJE
(NR_REZERWACJI)
ENABLE;
----------------------------------------------------------------------------------------------------
create or replace Procedure ZmienStatusRezerwacji(id_res REZERWACJE.NR_REZERWACJI%TYPE, stat
REZERWACJE.STATUS%TYPE) as
   canRes    integer;
   id_tour   int;
   count_res int;
BEGIN
   SELECT COUNT(*)
   INTO count_res
   FROM REZERWACJE
   WHERE REZERWACJE.NR_REZERWACJI = id_res;
   IF count_res = 0 THEN
       RAISE_APPLICATION_ERROR(-30067, 'Given reservation does not exist');
   END IF;

   IF stat != 'A' THEN
       select ID_WYCIECZKI
       INTO id_tour
       from REZERWACJE r
       where r.NR_REZERWACJI = id_res;

       select count(*)
       INTO canRes
       from WycieczkiDostepne wd
       where wd.ID_WYCIECZKI = id_tour;
       IF canRes = 0 THEN
           RAISE_APPLICATION_ERROR(-20011, 'Lack of free places');
       END IF;
   END IF;
↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓
↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓
 UPDATE REZERWACJE
      SET STATUS = stat
      WHERE NR_REZERWACJI = id_res;
      INSERT INTO REZERWACJE_LOG(ID_REZERWACJI, DATA, STATUS)
      VALUES (id_res, CURRENT_DATE, stat);
↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑
↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑
end;
```

7. Zmiana struktury bazy danych, w tabeli wycieczki dodajemy redundantne pole liczba_wolnych_miejsc
Należy zmodyfikować zestaw widoków. Proponuję dodać kolejne widoki (np. z sufiksem 2), które pobierają informację o wolnych miejscach z nowo dodanego pola.

Należy napisać procedurę przelicz która zaktualizuje wartość liczby wolnych miejsc dla już istniejących danych

Należy zmodyfikować warstwę procedur pobierających dane, podobnie jak w przypadku widoków.

Należy zmodyfikować procedury wprowadzające dane tak aby korzystały/aktualizowały pole liczba _wolnych_miejsc w tabeli wycieczki Najlepiej to zrobić tworząc nowe wersje (np. z sufiksem 2)

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

create view RezerwacjeWszystkie2 as
select w.ID_WYCIECZKI,
      w.NAZWA,
      w.kraj,
      w.data,

↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓
↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓
       w.LICZBA_WOLNYCH_MIEJSC,

↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑
↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑
       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
----------------------------------------------------------------------------------------------------
create view WycieczkiMiejsca2
as
select w.kraj,
      w.data,
      w.nazwa,
      w.LICZBA_MIEJSC,
      w.LICZBA_WOLNYCH_MIEJSC
from wycieczki w;
----------------------------------------------------------------------------------------------------
create view wycieczkidostepne2
as
select *
from WycieczkiMiejsca2
↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓
↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓
where liczba_wolnych_miejsc > 0
↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑
↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑
----------------------------------------------------------------------------------------------------
CREATE OR REPLACE PROCEDURE PrzeliczDane AS
```

```
BEGIN
   UPDATE WYCIECZKI w
   SET LICZBA_WOLNYCH_MIEJSC = LICZBA_MIEJSC -
                               (SELECT COUNT(*)
                                FROM REZERWACJE r
                                WHERE r.ID_WYCIECZKI = w.ID_WYCIECZKI
                                  AND r.STATUS <> 'A')
   WHERE ID_WYCIECZKI >= 0;
END;
-----------------------------------------------------------------------------------------------
CREATE OR REPLACE PROCEDURE
   dodaj_rezerwacje2(id_wycieczki int,
                     id_osoby int) AS
   exist integer;
BEGIN
   SELECT COUNT(*)
   INTO exist
   FROM OSOBY
   WHERE OSOBY.ID_OSOBY = dodaj_rezerwacje2.id_osoby;

   IF exist = 0
   THEN
       raise_application_error(-30009, 'The person with given id does not exist');
   END IF;

   SELECT COUNT(*)
   INTO exist
   FROM WYCIECZKIDOSTEPNE w
   WHERE w.ID_WYCIECZKI = dodaj_rezerwacje2.id_wycieczki;

   IF exist = 0
   THEN
       raise_application_error(-30008, 'The trip with given id does not exist');
   END IF;

   SELECT COUNT(*)
   INTO exist
   FROM REZERWACJE r
   WHERE r.ID_WYCIECZKI = dodaj_rezerwacje2.id_wycieczki
     AND r.ID_OSOBY = dodaj_rezerwacje2.id_osoby;

   IF exist > 0
   THEN
       raise_application_error(-30007, 'The reservation already exist');
   END IF;

   INSERT INTO REZERWACJE (id_wycieczki, id_osoby, STATUS)
   VALUES (dodaj_rezerwacje2.id_wycieczki, dodaj_rezerwacje2.id_osoby, 'N');
   UPDATE WYCIECZKI
   SET LICZBA_WOLNYCH_MIEJSC = LICZBA_WOLNYCH_MIEJSC - 1
   WHERE ID_WYCIECZKI = dodaj_rezerwacje2.id_wycieczki;

END dodaj_rezerwacje2;

-----------------------------------------------------------------------------------------------
create or replace Procedure ZmienStatusRezerwacji(id_res REZERWACJE.NR_REZERWACJI%TYPE, stat
REZERWACJE.STATUS%TYPE) as
   canRes    integer;
   id_tour   int;
```

```
   count_res int;
BEGIN
   SELECT COUNT(*)
   INTO count_res
   FROM REZERWACJE
   WHERE REZERWACJE.NR_REZERWACJI = id_res;
   IF count_res = 0 THEN
       RAISE_APPLICATION_ERROR(-30067, 'Given reservation does not exist');
   END IF;

   IF stat != 'A' THEN
       select ID_WYCIECZKI
       INTO id_tour
       from REZERWACJE r
       where r.NR_REZERWACJI = id_res;

       select count(*)
       INTO canRes
       from WycieczkiDostepne wd
       where wd.ID_WYCIECZKI = id_tour;
       IF canRes = 0 THEN
           RAISE_APPLICATION_ERROR(-20011, 'Lack of free places');
       END IF;
   END IF;

   UPDATE REZERWACJE
   SET STATUS = stat
   WHERE NR_REZERWACJI = id_res;

   INSERT INTO REZERWACJE_LOG(ID_REZERWACJI, DATA, STATUS)
   VALUES (id_res, CURRENT_DATE, stat);

end;
-----------------------------------------------------------------------------------------------
CREATE OR REPLACE PROCEDURE zmien_liczbe_miejsc2(id_w WYCIECZKI.ID_WYCIECZKI%TYPE,
                                                 nowe_miejsca WYCIECZKI.LICZBA_MIEJSC%TYPE) AS
   occupied integer;
   exist    integer;
   not_free integer;
BEGIN
   SELECT COUNT(*) INTO exist FROM WYCIECZKI WHERE WYCIECZKI.ID_WYCIECZKI = id_w;

   IF exist = 0 THEN
       raise_application_error(-25000, 'Given trip does not exist');
   END IF;

   BEGIN
       SELECT LICZBA_MIEJSC - LICZBA_WOLNYCH_MIEJSC
       INTO occupied
       FROM WYCIECZKIMIEJSCA wm
       WHERE wm.ID_WYCIECZKI = id_w;

       IF nowe_miejsca < 0 OR occupied > nowe_miejsca
       THEN
           raise_application_error(-20006, 'New places number is too low');
       end if;
       SELECT LICZBA_MIEJSC - LICZBA_WOLNYCH_MIEJSC
       INTO not_free
       FROM WYCIECZKI w
```

```
       WHERE w.ID_WYCIECZKI = id_w;

       IF nowe_miejsca < not_free THEN
           RAISE_APPLICATION_ERROR(-20015, 'New places number is less than reservations');
       END IF;

       UPDATE WYCIECZKI
       SET LICZBA_MIEJSC         = nowe_miejsca,
           LICZBA_WOLNYCH_MIEJSC = LICZBA_WOLNYCH_MIEJSC + (nowe_miejsca - LICZBA_MIEJSC)
       WHERE ID_WYCIECZKI = id_w;
   END;
End;
```

8. Zmiana strategii zapisywania do dziennika rezerwacji.
Realizacja przy pomocy triggerów Należy wprowadzić zmianę która spowoduje że zapis do dziennika rezerwacji będzie realizowany przy pomocy triggerów

triger obsługujący dodanie rezerwacji
triger obsługujący zmianę statusu
triger zabraniający usunięcia rezerwacji

Oczywiście po wprowadzeniu tej zmiany należy uaktualnić procedury modyfikujące dane. Najlepiej to zrobić tworząc nowe wersje (np. z sufiksem 3)

Dodawanie Triggerów

```
CREATE OR REPLACE TRIGGER ADD_RESERVATION_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 ADD_RESERVATION_TRIGGER;
-----------------------------------------------------------------------------------------------
CREATE OR REPLACE TRIGGER RESTRICT_DELETE_RESERVATION_TRIGGER
   BEFORE DELETE
   ON REZERWACJE
   FOR EACH ROW
BEGIN
   raise_application_error(-20008, 'Removing reservation forbidden.');
END  RESTRICT_DELETE_RESERVATION_TRIGGER;
-----------------------------------------------------------------------------------------------
CREATE OR REPLACE TRIGGER CHANGE_STATUS_TRIGGER
   AFTER UPDATE
   ON REZERWACJE
   FOR EACH ROW
DECLARE
   new_places INT;
BEGIN
   INSERT INTO REZERWACJE_LOG (ID_REZERWACJI, DATA, STATUS)
```

```
   VALUES (:NEW.NR_REZERWACJI, CURRENT_DATE, :NEW.STATUS);
END CHANGE_STATUS_TRIGGER;
```

9. Zmiana strategii obsługi redundantnego pola liczba_wolnych_miejsc . realizacja przy pomocy trigerów:

triger obsługujący dodanie rezerwacji
triger obsługujący zmianę statusu
triger obsługujący zmianę liczby miejsc na poziomie wycieczki

Oczywiście po wprowadzeniu tej zmiany należy uaktualnić procedury modyfikujące dane. Najlepiej to zrobić tworząc nowe wersje (np. z sufiksem 3)

```
CREATE OR REPLACE TRIGGER CHANGE_STATUS_TRIGGER2
   AFTER UPDATE
   ON REZERWACJE
   FOR EACH ROW
DECLARE
   new_places INT;
BEGIN
   IF :NEW.STATUS = 'A' THEN
       new_places := 1;
   ELSE
       new_places := 0;
   END IF;
   UPDATE WYCIECZKI w
   SET LICZBA_WOLNYCH_MIEJSC = LICZBA_WOLNYCH_MIEJSC + new_places
   WHERE w.ID_WYCIECZKI = :NEW.ID_WYCIECZKI;
END CHANGE_STATUS_TRIGGER2;
-------------------------------------------------------------------------------------------------
CREATE OR REPLACE TRIGGER PLACES_NUMBER_CHANGE_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;
```