

1 Tabele

1.1 OSOBY

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

	ID_OSOBY	IMIE	NAZWISKO	PESEL	KONTAKT
1	1	Adam	Kowalski	87654321	tel: 6623
2	2	Jan	Nowak	12345678	tel: 2312, dzwonić po 18.00





1.2 WYCIECZKI

```
CREATE OR REPLACE 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
);
```

	ID_WYCIECZKI	NAZWA	KRAJ	DATA	OPIS	LICZBA_MIEJSC
1	22	Helsinki	USA	2018-10-22 00:00:00	super wyjazd ...	5
2	1	Piękny Kraków	Polska	2017-02-03 00:00:00	Najciekawa wycieczka ...	2
3	2	Wieliczka	Polska	2017-03-03 00:00:00	Zadziwiająca kopalnia ...	2
4	3	Nowy Jork	USA	2019-03-03 00:00:00	Świetny wyjazd ...	5
5	4	Warszawy	USA	2018-10-13 00:00:00	Świetny wyjazd ...	5
6	5	Wieliczka2	Polska	2017-03-03 00:00:00	Zadziwiająca kopalnia ...	2

1.3 REZERWACJE

```
CREATE OR REPLACE 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
);
```

	 NR_REZERWACJI ▾	 ID_WYCIECZKI ▾	 ID_OSOBY ▾	 STATUS ▾
1	1	1	1	N
2	2	2	2	P
3	4	5	1	P
4	6	3	2	N
5	7	22	2	N

2 Widoki

2.1 WYCIECZKI OSOBY

```
CREATE VIEW WYCIECZKI_OSOBY AS
SELECT A.KRAJ, A.DATA, A.NAZWA, C.IMIE, C.NAZWISKO, B.STATUS
FROM WYCIECZKI A
     INNER JOIN REZERWACJE B ON A.ID_WYCIECZKI = B.ID_WYCIECZKI
     INNER JOIN OSOBY C ON B.ID_OSOBY = C.ID_OSOBY
```

	KRAJ	DATA	NAZWA	IMIE	NAZWISKO	STATUS
1	USA	2018-10-22 00:00:00	Helsinki	Jan	Nowak	N
2	Polska	2017-02-03 00:00:00	Piękny Kraków	Adam	Kowalski	N
3	Polska	2017-03-03 00:00:00	Wieliczka	Jan	Nowak	P
4	USA	2019-03-03 00:00:00	Nowy Jork	Jan	Nowak	N
5	Polska	2017-03-03 00:00:00	Wieliczka2	Adam	Kowalski	P
6	Polska	2017-03-03 00:00:00	Wieliczka2	Jan	Nowak	P
7	Polska	2017-03-03 00:00:00	Wieliczka2	Jan	Nowak	N

2.2 WYCIECZKI OSOBY POTWIERDZONE

```
CREATE VIEW WYCIECZKI_OSOBY_POTWIERDZONE AS
SELECT W.KRAJ, W.DATA, W.NAZWA, 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 R.ID_OSOBY = O.ID_OSOBY
WHERE R.STATUS = 'P'
```

	KRAJ	DATA	NAZWA	IMIE	NAZWISKO	STATUS
1	Polska	2017-03-03 00:00:00	Wieliczka2	Adam	Kowalski	P
2	Polska	2017-03-03 00:00:00	Wieliczka	Jan	Nowak	P
3	Polska	2017-03-03 00:00:00	Wieliczka2	Jan	Nowak	P

2.3 WYCIECZKI PRZYSZLE

```
CREATE VIEW WYCIECZKI_PRZYSZLE AS
SELECT W.KRAJ, W.DATA, W.NAZWA, 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 R.ID_OSOBY = O.ID_OSOBY
WHERE W.DATA > (SELECT CURRENT_DATE FROM DUAL)
```

	KRAJ	DATA	NAZWA	IMIE	NAZWISKO	STATUS
1	USA	2019-03-03 00:00:00	Nowy Jork	Jan	Nowak	N
2	USA	2018-10-22 00:00:00	Helsinki	Jan	Nowak	N

2.4 WYCIECZKI MIEJSCA

```
CREATE VIEW WYCIECZKI_MIEJSCA AS
SELECT W.KRAJ,
       W.DATA,
       W.NAZWA,
       W.LICZBA_MIEJSC,
       W.LICZBA_MIEJSC -
       NVL((SELECT COUNT(*) FROM REZERWACJE R WHERE W.ID_WYCIECZKI = R.ID_WYCIECZKI GROUP BY R.ID_
            0) AS LICZBA_MIEJSC_WOLNYCH
FROM WYCIECZKI W
```

	KRAJ	DATA	NAZWA	LICZBA_MIEJSC	LICZBA_MIEJSC_WOLNYCH
1	USA	2018-10-22 00:00:00	Helsinki	5	4
2	USA	2018-10-13 00:00:00	Warszawy	5	5
3	USA	2019-03-03 00:00:00	Nowy Jork	5	4
4	Polska	2017-02-03 00:00:00	Piękny Kraków	2	1
5	Polska	2017-03-03 00:00:00	Wieliczka	2	1
6	Polska	2017-03-03 00:00:00	Wieliczka2	2	1

2.5 DOSTĘPNE WYCIEZKI

```
CREATE VIEW DOSTĘPNE_WYCIEZKI AS
SELECT W.KRAJ,
       W.DATA,
       W.NAZWA,
       W.LICZBA_MIEJSC,
       W.LICZBA_MIEJSC -
       NVL((SELECT COUNT(*) FROM REZERWACJE R
            WHERE W.ID_WYCIEZKI = R.ID_WYCIEZKI GROUP BY R.ID_WYCIEZKI),
            0) AS LICZBA_MIEJSC_WOLNYCH
FROM WYCIEZKI W
WHERE W.LICZBA_MIEJSC -
      NVL((SELECT COUNT(*) FROM REZERWACJE R WHERE W.ID_WYCIEZKI = R.ID_WYCIEZKI GROUP BY R.ID_WYCIEZKI),
          0) > 0
      AND (SELECT CURRENT_DATE FROM DUAL) < W.DATA
```

	KRAJ	DATA	NAZWA	LICZBA_MIEJSC	LICZBA_MIEJSC_WOLNYCH
1	USA	2019-03-03 00:00:00	Nowy Jork	5	4

2.6 REZERWACJE DO ANULOWANIA

```
CREATE VIEW REZERWACJE_DO_ANULOWANIA AS
SELECT R.NR_REZERWACJI AS NR_REZERWACJI_DO_ANULOWANIA
FROM REZERWACJE R
      INNER JOIN WYCIEZKI W ON W.ID_WYCIEZKI = R.ID_WYCIEZKI
WHERE R.STATUS = 'N'
      AND W.DATA - (SELECT CURRENT_DATE FROM DUAL) < 7
      AND W.DATA > (SELECT CURRENT_DATE FROM DUAL)
```

	NR_REZERWACJI_DO_ANULOWANIA
1	7

3 Funkcje

3.1 UCZESTNICY WYCIECZKI

uczestnicy_wycieczki(id_wycieczki), procedura ma zwracać podobny zestaw danych jak widok wycieczki_osoby.

3.1.1 TYPE UCZESTNICY WYCIECZKI

```
CREATE OR REPLACE TYPE TYPE_UCZESTNICY_WYCIECZKI AS OBJECT (  
    ID_OSOBY NUMBER,  
    IMIE      VARCHAR(50),  
    NAZWISKO VARCHAR(50)  
);
```

3.1.2 TABLICA UCZESTNICY WYCIECZKI

```
CREATE OR REPLACE TYPE TAB_UCZESTNICY_WYCIECZKI AS TABLE OF TYPE_UCZESTNICY_WYCIECZKI;
```

3.1.3 FUNKCJA UCZESTNICY WYCIECZKI

```
CREATE OR REPLACE FUNCTION UCZESTNICY_WYCIECZKI(ID IN NUMBER)  
    RETURN TAB_UCZESTNICY_WYCIECZKI PIPELINED  
AS  
BEGIN  
    FOR X IN (SELECT R.ID_OSOBY, O.IMIE, O.NAZWISKO, R.ID_WYCIECZKI  
                FROM REZERWACJE R  
                INNER JOIN OSOBY O ON O.ID_OSOBY = R.ID_OSOBY  
                WHERE R.ID_WYCIECZKI = ID)  
    LOOP  
        PIPE ROW (TYPE_UCZESTNICY_WYCIECZKI(X.ID_OSOBY, X.IMIE, X.NAZWISKO));  
    END LOOP;  
END;
```

	"UCZESTNICY_WYCIECZKI(5)"
1	{{1,Adam,Kowalski}}

3.2 REZERWACJA OSOBY

rezerwacje_osoby(id_osoby), procedura ma zwracać podobny zestaw danych jak widok wycieczki_osoby

3.2.1 TYPE REZERWACJE OSOBY

```
CREATE OR REPLACE TYPE TYPE_REZERWACJE_OSOBY AS OBJECT (  
    NR_REZERWACJI NUMBER  
);
```

3.2.2 TABLICA REZERWACJE OSOBY

```
CREATE OR REPLACE TYPE TAB_REZERWACJE_OSOBY AS TABLE OF TYPE_REZERWACJE_OSOBY;
```

3.2.3 FUNKCJA REZERWACJE OSOBY

```
CREATE OR REPLACE FUNCTION REZERWACJE_OSOBY(ID NUMBER)  
    RETURN TAB_REZERWACJE_OSOBY PIPELINED  
AS  
BEGIN  
    FOR X IN (SELECT R.NR_REZERWACJI  
                FROM REZERWACJE R  
                INNER JOIN OSOBY O ON R.ID_OSOBY = O.ID_OSOBY  
                WHERE O.ID_OSOBY = ID)
```

```

LOOP
    PIPE ROW (TYPE_REZERWACJE_OSOBY(X.NR_REZERWACJI));
END LOOP;
END;

```

"REZERWACJE_OSOBY(2)"
1 {{2},{6},{7}}

Pomocniczo tabela rezerwacji:

	NR_REZERWACJI	ID_WYCIECZKI	ID_OSOBY	STATUS
1	1	1	1	N
2	2	2	2	P
3	4	5	1	P
4	6	3	2	N
5	7	22	2	N

3.3 PRZYSZŁE REZERWACJE OSOBY

przyszle_rezerwacje_osoby(id_osoby)

```

CREATE OR REPLACE FUNCTION PRZYSZLE_REZERWACJE_OSOBY(ID NUMBER)
RETURN TAB_REZERWACJE_OSOBY PIPELINED
AS
BEGIN
    FOR X IN (SELECT R.NR_REZERWACJI
              FROM REZERWACJE R
                   INNER JOIN OSOBY O ON R.ID_OSOBY = O.ID_OSOBY
                   INNER JOIN WYCIECZKI W ON R.ID_WYCIECZKI = W.ID_WYCIECZKI
              WHERE O.ID_OSOBY = ID AND W.DATA > (SELECT CURRENT_DATE FROM DUAL)
            )
    LOOP
        PIPE ROW (TYPE_REZERWACJE_OSOBY(X.NR_REZERWACJI));
    END LOOP;
END;

```

"PRZYSZLE_REZERWACJE_OSOBY(2)"
1 {{6},{7}}

Pomocniczo tabela rezerwacji:

	NR_REZERWACJI	ID_WYCIECZKI	ID_OSOBY	STATUS
1	1	1	1	N
2	2	2	2	P
3	4	5	1	P
4	6	3	2	N
5	7	22	2	N

Pomocniczo tabela wycieczki:

	ID_WYCIECZKI	NAZWA	KRAJ	DATA	OPIS	LICZBA_MIEJSC
1	22	Helsinki	USA	2018-10-22 00:00:00	super wyjazd ...	5
2	1	Piękny Kraków	Polska	2017-02-03 00:00:00	Najciekawa wycieczka ...	2
3	2	Wieliczka	Polska	2017-03-03 00:00:00	Zadziwiająca kopalnia ...	2
4	3	Nowy Jork	USA	2019-03-03 00:00:00	Świetny wyjazd ...	5
5	4	Warszawy	USA	2018-10-13 00:00:00	Świetny wyjazd ...	5
6	5	Wieliczka2	Polska	2017-03-03 00:00:00	Zadziwiająca kopalnia ...	2

3.4 DOSTĘPNE WYCIECZKI

dostepne_wycieczki(kraj, data_od, data_do)

3.4.1 TYPE DOSTĘPNE WYCIECZKI

```
CREATE OR REPLACE TYPE TYPE_DOSTEPNE_WYCIECZKI AS OBJECT (  
    KRAJ VARCHAR(50),  
    DATA DATE  
);
```

3.4.2 TABLICA DOSTĘPNE WYCIECZKI

```
CREATE OR REPLACE TYPE TAB_DOSTEPNE_WYCIECZKI AS TABLE OF TYPE_DOSTEPNE_WYCIECZKI;
```

3.4.3 FUNKCJA DOSTĘPNE WYCIECZKI

```
CREATE OR REPLACE FUNCTION DOSTEPNE_WYCIECZKI(KRAJ VARCHAR2, DATA_OD DATE, DATA_DO DATE)  
    RETURN TAB_DOSTEPNE_WYCIECZKI PIPELINED  
AS  
    BEGIN  
        FOR X IN (SELECT W.KRAJ, W.DATA  
                    FROM WYCIECZKI W  
                    WHERE W.LICZBA_MIEJSC -  
                        NVL(  
                            (SELECT COUNT(*) FROM REZERWACJE R WHERE W.ID_WYCIECZKI = R.ID_WYCIECZKI GROUP  
                                BY W.ID_WYCIECZKI)  
                            , 0)  
                        > 0  
                    AND ((SELECT CURRENT_DATE FROM DUAL) < W.DATA)  
                    AND (W.DATA BETWEEN DATA_OD AND DATA_DO)  
                    AND W.KRAJ = KRAJ)  
        LOOP  
            PIPE ROW (TYPE_DOSTEPNE_WYCIECZKI(X.KRAJ, X.DATA));  
        END LOOP;  
        RETURN;  
    END;
```

```
DOSTEPNE_WYCIECZKI('USA',TO_DATE('2018-10-01','YYYY-MM-DD'),TO_DATE('2020-10-22','YYYY-MM-DD'))  
1 {{USA,2018-10-22 00:00:00},{USA,2019-03-03 00:00:00}}
```

4 Procedury

4.1 DOSTĘPNE MIEJSCA

Zwraca liczbę dostępnych miejsc dla podanego id_wycieczki.

```
CREATE OR REPLACE FUNCTION DOSTEPNE_MIEJSCA(ID_W NUMBER)
RETURN NUMBER
IS
    LICZBA_WOLNYCH_MIEJSC NUMBER;
BEGIN
    SELECT W.LICZBA_MIEJSC -
        NVL((SELECT COUNT(*) FROM REZERWACJE R WHERE W.ID_WYCIECZKI = R.ID_WYCIECZKI GROUP BY R.ID_WYCIECZKI)
        ) INTO LICZBA_WOLNYCH_MIEJSC
    FROM WYCIECZKI W
    WHERE W.ID_WYCIECZKI = 6;
    RETURN LICZBA_WOLNYCH_MIEJSC;
END;
```

4.2 DODAJ REZERWACJE

dodaj_rezerwacje(id_wycieczki, id_osoby), procedura powinna kontrolować czy wycieczka jeszcze się nie odbyła, i czy są wolne miejsca.

```
CREATE OR REPLACE PROCEDURE DODAJ_REZERWACJE(ID_W NUMBER, ID_O NUMBER)
AS
BEGIN
    DECLARE
        WYCIECZKI NUMBER;
        OSOBA      NUMBER;
    BEGIN





        SELECT COUNT(W.ID_WYCIECZKI) INTO WYCIECZKI FROM WYCIECZKI W WHERE W.ID_WYCIECZKI = ID_W;
        SELECT O.ID_OSOBY INTO OSOBA FROM OSOBY O WHERE O.ID_OSOBY = ID_O;

        IF WYCIECZKI > 0 AND OSOBA > 0 AND DOSTEPNE_MIEJSCA(ID_W) > 0
        THEN
            INSERT INTO REZERWACJE (ID_WYCIECZKI, ID_OSOBY, STATUS) VALUES (ID_W, ID_O, 'N');
        END IF;

    END;
END;
```

Po wykonaniu kodu

```
begin
    DODAJ_REZERWACJE(3, 1);
end;
```

	 NR_REZERWACJI	 ID_WYCIECZKI	 ID_OSOBY	 STATUS
1	21	3	1	N
2	1	1	1	N
3	2	2	2	P
4	4	5	1	P
5	6	3	2	N
6	7	22	2	N

Pojawia się nowa rezerwacja, ponieważ wycieczka o numerze id = 3 posiada wolne miejsca oraz jej data rozpoczęcia jest większa od dzisiejszej daty.

4.3 ZMIEN STATUS REZERWACJI





zmien_status_rezerwacji(id_rezerwacji, status), procedura kontrolować czy możliwa jest zmiana statusu, np. zmiana statusu już anulowanej wycieczki (przywrócenie do stanu aktywnego nie zawsze jest możliwe)

```
create procedure zmien_status_rezerwacji(id_rezerwacji NUMBER, nowy_status_ char)
as
begin
  declare
    id_r          NUMBER;
    s              CHAR;
    dzisiejsza_data DATE;
  begin
    select count(r.NR_REZERWACJI) into id_r from REZERWACJE r where r.NR_REZERWACJI = id_rezerwacji;
    select r.status into s from REZERWACJE r where r.NR_REZERWACJI = id_rezerwacji;
    select current_date into dzisiejsza_data from dual;
    if id_r = 1
    then
      if (s <> 'A') and nowy_status_ in ('N', 'P', 'Z') and nowy_status_ <> s
      then
        update REZERWACJE r set r.STATUS = nowy_status_ where r.NR_REZERWACJI = id_rezerwacji;
      end if;
    end if;
  end;
end;
```

Po wykonaniu procedury:

```
begin
  ZMIEN_STATUS_REZERWACJI(7, 'P');
end;
```

Następuje zmiana statusu rezerwacji dla rezerwacji o id = 7:

	 NR_REZERWACJI	 ID_WYCIECZKI	 ID_OSOBY	 STATUS
1	21	3	1	N
2	1	1	1	N
3	2	2	2	P
4	4	5	1	P
5	6	3	2	N
6	7	22	2	P

4.4 ZMIEN LICZBE MIEJSC

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

```
CREATE OR REPLACE PROCEDURE ZMIEN_LICZBE_MIEJSC(ID_WYCIECZKI_ NUMBER, NOWA_LICZBA_MIEJSC NUMBER)
AS
BEGIN
  DECLARE
    CALKOWITA_LICZBA_MIEJSC NUMBER;
    DOSTEPNE_MIEJSCA_ NUMBER;
  BEGIN
    SELECT W.LICZBA_MIEJSC INTO CALKOWITA_LICZBA_MIEJSC FROM WYCIECZKI W WHERE W.ID_WYCIECZKI = ID_WYCIECZKI_;
    SELECT DOSTEPNE_MIEJSCA(ID_WYCIECZKI_) INTO DOSTEPNE_MIEJSCA_ FROM DUAL;
    IF (NOWA_LICZBA_MIEJSC >= (CALKOWITA_LICZBA_MIEJSC - DOSTEPNE_MIEJSCA_))
    THEN
      UPDATE WYCIECZKI W SET W.LICZBA_MIEJSC = NOWA_LICZBA_MIEJSC WHERE W.ID_WYCIECZKI = ID_WYCIECZKI_;
    END IF;
  END;
```

```
END;
END;
```

Po wykonaniu procedury:

```
begin
  ZMIEN_LICZBE_MIEJSC(5, 7);
end;
```

Następuje zmiana miejsc dla wycieczki o id = 5:

	ID_WYCIEZKI	NAZWA	KRAJ	DATA	OPIS	LICZBA_MIEJSC
1	22	Helsinki	USA	2018-10-22 00:00:00	super wyjazd ...	5
2	1	Piękny Kraków	Polska	2017-02-03 00:00:00	Najciekawa wycieczka ...	2
3	2	Wieliczka	Polska	2017-03-03 00:00:00	Zadziwiająca kopalnia ...	2
4	3	Nowy Jork	USA	2019-03-03 00:00:00	Świetny wyjazd ...	5
5	4	Warszawy	USA	2018-10-13 00:00:00	Świetny wyjazd ...	5
6	5	Wieliczka2	Polska	2017-03-03 00:00:00	Zadziwiająca kopalnia ...	7

5 Polecenie 6

5.1 ZMIEN STATUS REZERWACJI ORAZ TABELA REZERWACJE LOG

Dodajemy tabelę dziennikującą zmiany statusu rezerwacji rezerwacje_log(id, id_rezerwacji, data, status)
Należy zmienić warstwę procedur modyfikujących dane tak aby dopisywały informację do dziennika.
Przez pomyłkę zaimplementowałem nazwę dziennik_rezerwacji zamiast rezerwacje_log. Mam nadzieję że nie stanowi to problemu.

5.1.1 TABELA DZIENNIK REZERWACJI

```
CREATE TABLE DZIENNIK_REZERWACJI
(
ID_ZMIANY_STATUSU INT GENERATED ALWAYS AS IDENTITY NOT NULL
, NR_REZERWACJI INT
, DATA DATE
, NOWY_STATUS CHAR
, CONSTRAINT DZIENNIK_REZERWACJI_PK PRIMARY KEY
(
ID_ZMIANY_STATUSU
)
ENABLE
);
```

	ID_ZMIANY_STATUSU	NR_REZERWACJI	DATA	NOWY_STATUS
1	41	7	2018-10-17 17:44:52	P
2	42	6	2018-10-17 17:53:24	P
3	21	19	2018-10-15 11:37:02	N
4	22	20	2018-10-15 11:37:02	P
5	23	21	2018-10-15 11:37:03	P
6	24	22	2018-10-15 11:37:03	P
7	25	23	2018-10-15 11:37:03	N
8	26	24	2018-10-15 11:37:03	N
9	27	25	2018-10-15 11:52:44	N
10	28	26	2018-10-15 11:52:44	P
11	29	27	2018-10-15 11:52:44	P
12	30	28	2018-10-15 11:52:45	P
13	31	29	2018-10-15 11:52:45	N
14	32	30	2018-10-15 11:52:45	N
15	33	30	2018-10-15 12:03:35	Z

5.1.2 ZMIEN STATUS REZERWACJI

```
CREATE OR REPLACE PROCEDURE ZMIEN_STATUS_REZERWACJI_2(ID_REZERWACJI NUMBER, NOWY_STATUS_ CHAR)
AS
BEGIN
    DECLARE
        ID_R          NUMBER;
        S              CHAR;
        DZISIEJSZA_DATA DATE;
    BEGIN
        SELECT COUNT(R.NR_REZERWACJI) INTO ID_R FROM REZERWACJE R WHERE R.NR_REZERWACJI = ID_REZERWACJI;
        SELECT R.STATUS INTO S FROM REZERWACJE R WHERE R.NR_REZERWACJI = ID_REZERWACJI;
        SELECT CURRENT_DATE INTO DZISIEJSZA_DATA FROM DUAL;
        IF ID_R = 1
        THEN
            IF (S <> 'A') AND NOWY_STATUS_ IN ('N', 'P', 'Z') AND NOWY_STATUS_ <> S
            THEN
                UPDATE REZERWACJE R SET R.STATUS = NOWY_STATUS_ WHERE R.NR_REZERWACJI = ID_REZERWACJI;
                INSERT INTO DZIENNIK_REZERWACJI (NR_REZERWACJI, DATA, NOWY_STATUS)
                VALUES (ID_REZERWACJI, DZISIEJSZA_DATA, NOWY_STATUS_);
            END IF;
        END IF;
    END;
```

```

        END IF;
    END;
END;

```

Po wykonaniu procedury:

```

begin
    ZMIEN_STATUS_REZERWACJI_2(6, 'Z');
end;

```

Następuje zmiana statusu rezerwacji dla rezerwacji o id = 6 i zostaje to zapisane do dziennika rezerwacji.

	ID_ZMIANY_STATUSU	NR_REZERWACJI	DATA	NOWY_STATUS
1	42	6	2018-10-17 17:53:24	P

6 Polecenie 7

6.1 ZMODYFIKOWANA TABELA WYCIECZKI

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

```
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
  ,LICZBA_WOLNYCH_MIEJSC INT
  ,CONSTRAINT WYCIECZKI_PK PRIMARY KEY
    (
      ID_WYCIECZKI
    )
  ENABLE
);
```

	ID_WYCIECZKI	NAZWA	KRAJ	DATA	OPIS	LICZBA_MIEJSC	LICZBA_WOLNYCH_MIEJSC
1	22	Helsinki	USA	2018-10-22 00:00:00	super wyjazd ...	5	5
2	1	Piękny Kraków	Polska	2017-02-03 00:00:00	Najciekawa wycieczka ...	2	1
3	2	Wieliczka	Polska	2017-03-03 00:00:00	Zadziwiająca kopalnia ...	2	1
4	3	Nowy Jork	USA	2019-03-03 00:00:00	Świetny wyjazd ...	5	2
5	4	Warszawy	USA	2018-10-13 00:00:00	Świetny wyjazd ...	5	5
6	5	Wieliczka2	Polska	2017-03-03 00:00:00	Zadziwiająca kopalnia ...	7	1

6.2 DOSTĘPNE WYCIECZKI 2

Należy zmodyfikować zestaw widoków. Proponuję dodać kolejne widoki (np. z sufiksem 2), które pobierają informację o wolnych miejscach z nowo dodanego pola.

```
CREATE VIEW DOSTEPNE_WYCIECZKI_2 AS
SELECT *
FROM WYCIECZKI W
WHERE W.LICZBA_WOLNYCH_MIEJSC > 0
AND W.DATA > (SELECT CURRENT_DATE FROM DUAL)
```

	ID_WYCIECZKI	NAZWA	KRAJ	DATA	OPIS	LICZBA_MIEJSC	LICZBA_WOLNYCH_MIEJSC
1	22	Helsinki	USA	2018-10-22 00:00:00	super wyjazd ...	5	5
2	3	Nowy Jork	USA	2019-03-03 00:00:00	Świetny wyjazd ...	5	2

6.3 PRZELICZ

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

```
CREATE OR REPLACE PROCEDURE PRZELICZ AS
BEGIN
  DECLARE
    VAL NUMBER;
  BEGIN
    FOR REC IN (SELECT W.ID_WYCIECZKI, W.LICZBA_MIEJSC -
      NVL((SELECT COUNT(*)
        FROM REZERWACJE R
        WHERE W.ID_WYCIECZKI = R.ID_WYCIECZKI
        GROUP BY R.ID_WYCIECZKI),
      0) LICZBA_MIEJSC_WOLNYCH
      FROM WYCIECZKI W)
    LOOP
      UPDATE WYCIECZKI S
```

```

        SET S.LICZBA_WOLNYCH_MIEJSC = REC.LICZBA_MIEJSC_WOLNYCH
        WHERE S.ID_WYCIECZKI = REC.ID_WYCIECZKI;
    END LOOP;
END;
END;

```

Sprawdzenie procedury przelicz następuje dalej.

6.4 DOSTEPNE MIEJSCA 2

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

```

CREATE OR REPLACE FUNCTION DOSTEPNE_MIEJSCA_2(ID_W NUMBER)
RETURN NUMBER
IS
    LICZBA_WOLNYCH_MIEJSC_ NUMBER;
BEGIN
    SELECT W.LICZBA_WOLNYCH_MIEJSC INTO LICZBA_WOLNYCH_MIEJSC_ FROM WYCIECZKI W WHERE W.ID_WYCIECZKI = ID_W;
    RETURN LICZBA_WOLNYCH_MIEJSC_;
END;

```

	"DOSTEPNE_MIEJSCA_2(5)"	
1		1

6.5 DOSTEPNE WYCIECZKI 2

```

CREATE OR REPLACE FUNCTION DOSTEPNE_WYCIECZKI_2_(KRAJ_ VARCHAR2, DATA_OD DATE, DATA_DO DATE)
RETURN TAB_DOSTEPNE_WYCIECZKI PIPELINED
AS
BEGIN
    FOR X IN (SELECT W.KRAJ, W.DATA
              FROM WYCIECZKI W
              WHERE W.LICZBA_WOLNYCH_MIEJSC > 0
                AND (SELECT CURRENT_DATE FROM DUAL) < W.DATA
                AND W.DATA BETWEEN DATA_OD AND DATA_DO
                AND W.KRAJ = KRAJ_)
    LOOP
        PIPE ROW (TYPE_DOSTEPNE_WYCIECZKI(X.KRAJ, X.DATA));
    END LOOP;
    RETURN;
END;

```

	DOSTEPNE_WYCIECZKI_2_('USA',TO_DATE('2018-03-22','YYYY-MM-DD'),TO_DATE('2020-10-22','YYYY-MM-DD'))	
1	{{USA,2018-10-22 00:00:00},{USA,2019-03-03 00:00:00}}	

6.6 DODAJ REZERWACJE 2

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

```

CREATE OR REPLACE PROCEDURE DODAJ_REZERWACJE_2(ID_W NUMBER, ID_O NUMBER)
AS
BEGIN
    DECLARE
        WYCIECZKI NUMBER;
        OSOBA      NUMBER;
    BEGIN
        SELECT COUNT(W.ID_WYCIECZKI) INTO WYCIECZKI FROM WYCIECZKI W WHERE W.ID_WYCIECZKI = ID_W;
    END;
END;

```

```

SELECT O.ID_OSOBY INTO OSOBA FROM OSOBY O WHERE O.ID_OSOBY = ID_O;

IF WYCIECZKI > 0 AND OSOBA > 0
THEN
    INSERT INTO REZERWACJE (ID_WYCIECZKI, ID_OSOBY, STATUS) VALUES (ID_W, ID_O, 'N');
    BEGIN
        PRZELICZ();
    END;
END IF;
END;
END;

```

	ID_WYCIECZKI	NAZWA	KRAJ	DATA	OPIS	LICZBA_MIEJSC	LICZBA_WOLNYCH_MIEJSC
1	22	Helsinki	USA	2018-10-22 00:00:00	super wyjazd ...	5	4
2	1	Piękny Kraków	Polska	2017-02-03 00:00:00	Najciekawa wycieczka ...	2	1
3	2	Wieliczka	Polska	2017-03-03 00:00:00	Zadziwiająca kopalnia ...	2	1
4	3	Nowy Jork	USA	2019-03-03 00:00:00	Świetny wyjazd ...	5	3
5	4	Warszawy	USA	2018-10-13 00:00:00	Świetny wyjazd ...	5	5
6	5	Wieliczka2	Polska	2017-03-03 00:00:00	Zadziwiająca kopalnia ...	7	6

```

begin
    DODAJ_REZERWACJE_2(5, 2);
end;

```

Po wykonaniu procedury:

	ID_WYCIECZKI	NAZWA	KRAJ	DATA	OPIS	LICZBA_MIEJSC	LICZBA_WOLNYCH_MIEJSC
1	22	Helsinki	USA	2018-10-22 00:00:00	super wyjazd ...	5	4
2	1	Piękny Kraków	Polska	2017-02-03 00:00:00	Najciekawa wycieczka ...	2	1
3	2	Wieliczka	Polska	2017-03-03 00:00:00	Zadziwiająca kopalnia ...	2	1
4	3	Nowy Jork	USA	2019-03-03 00:00:00	Świetny wyjazd ...	5	3
5	4	Warszawy	USA	2018-10-13 00:00:00	Świetny wyjazd ...	5	5
6	5	Wieliczka2	Polska	2017-03-03 00:00:00	Zadziwiająca kopalnia ...	7	5

Uaktualnione zostało pole LICZBA_WOLNYCH_MIEJSC, co potwierdza poprawność działania procedury przelicz. Tutaj wynik z tabeli REZERWACJE:

	NR_REZERWACJI	ID_WYCIECZKI	ID_OSOBY	STATUS
1	1	1	1	N
2	2	2	2	P
3	4	5	1	P
4	6	3	2	P
5	7	22	2	P
6	21	3	1	N
7	41	5	2	N

6.7 ZMIEN LICZBE MIEJSC 2

```

CREATE PROCEDURE ZMIEN_LICZBE_MIEJSC_2(ID_WYCIECZKI_ NUMBER, NOWA_LICZBA_MIEJSC NUMBER)
AS
BEGIN
    DECLARE
        CALKOWITA_LICZBA_MIEJSC NUMBER;
        DOSTEPNE_MIEJSCA_        NUMBER;
    BEGIN
        SELECT W.LICZBA_MIEJSC INTO CALKOWITA_LICZBA_MIEJSC FROM WYCIECZKI W WHERE W.ID_WYCIECZKI = ID_WYCIECZKI_;
        SELECT DOSTEPNE_MIEJSCA(ID_WYCIECZKI_) INTO DOSTEPNE_MIEJSCA_ FROM DUAL;
        IF (NOWA_LICZBA_MIEJSC >= (CALKOWITA_LICZBA_MIEJSC - DOSTEPNE_MIEJSCA_))
        THEN
            UPDATE WYCIECZKI W SET W.LICZBA_MIEJSC = NOWA_LICZBA_MIEJSC WHERE W.ID_WYCIECZKI = ID_WYCIECZKI_;
            BEGIN
                PRZELICZ();
            END;
        END IF;
    END;
END;
END;

```

Po wykonaniu procedury:

```
begin
  ZMIEN_LICZBE_MIEJSC_2(5, 5);
end;
```

	ID_WYCIEZKI	NAZWA	KRAJ	DATA	OPIS	LICZBA_MIEJSC	LICZBA_WOLNYCH_MIEJSC
1	22	Hel sinki	USA	2018-10-22 00:00:00	super wyjazd ...	5	4
2	1	Piękny Kraków	Polska	2017-02-03 00:00:00	Najciekawa wycieczka ...	2	1
3	2	Wieliczka	Polska	2017-03-03 00:00:00	Zadziwiająca kopalnia ...	2	1
4	3	Nowy Jork	USA	2019-03-03 00:00:00	Świetny wyjazd ...	5	3
5	4	Warszawy	USA	2018-10-13 00:00:00	Świetny wyjazd ...	5	5
6	5	Wieliczka2	Polska	2017-03-03 00:00:00	Zadziwiająca kopalnia ...	5	3

7 Polecenie 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

7.1 TRIGER DODANIE REZERWACJI

trigger obsługujący dodanie rezerwacji

```
CREATE OR REPLACE TRIGGER DODANIE_REZERWACJI
AFTER INSERT
ON REZERWACJE
FOR EACH ROW
DECLARE
    AKTUALNA_DATA DATE;
BEGIN
    SELECT CURRENT_DATE INTO AKTUALNA_DATA FROM DUAL;
    INSERT INTO DZIENNIK_REZERWACJI (NR_REZERWACJI, DATA, NOWY_STATUS)
    VALUES (:NEW.NR_REZERWACJI, AKTUALNA_DATA, :NEW.STATUS);
END;
```

Po dodaniu rezerwacji kodem poniżej:

```
begin
    ZMIEN_STATUS_REZERWACJI_3(41, 'Z');
end;
```

Dziennik rezerwacji:

	ID_ZMIANY_STATUSU	NR_REZERWACJI	DATA	NOWY_STATUS
1	61	41	2018-10-18 15:39:56	Z
2	62	61	2018-10-18 15:53:32	N

Rezerwacje:

	NR_REZERWACJI	ID_WYCIEZKI	ID_OSOBY	STATUS
1	21	3	1	N
2	61	3	2	N
3	1	1	1	N
4	2	2	2	P
5	41	5	2	Z
6	4	5	1	P
7	6	3	2	P

7.2 TRIGER ZMIANA STATUSU

trigger obsługujący zmianę statusu

```
CREATE OR REPLACE TRIGGER ZMIANA_STATUSU_REZERWACJI
AFTER UPDATE
ON REZERWACJE
FOR EACH ROW
DECLARE
    AKTUALNA_DATA DATE;
BEGIN
    SELECT CURRENT_DATE INTO AKTUALNA_DATA FROM DUAL;
    INSERT INTO DZIENNIK_REZERWACJI (NR_REZERWACJI, DATA, NOWY_STATUS)
    VALUES (:NEW.NR_REZERWACJI, AKTUALNA_DATA, :NEW.STATUS);
END;
```

	NR_REZERWACJI	ID_WYCIEZKI	ID_OSOBY	STATUS
1	21	3	1	N
2	1	1	1	N
3	2	2	2	P
4	41	5	2	N
5	4	5	1	P
6	6	3	2	P
7	7	22	2	P

Po zmianie statusu rezerwacji metodą z sufiksem 3 - przepis podany poniżej.

```
begin
  ZMIEN_STATUS_REZERWACJI_3(41, 'Z');
end;
```

Rezerwacje:

	NR_REZERWACJI	ID_WYCIEZKI	ID_OSOBY	STATUS
1	21	3	1	N
2	1	1	1	N
3	2	2	2	P
4	41	5	2	Z
5	4	5	1	P
6	6	3	2	P
7	7	22	2	P

Dziennik rezerwacji:

	ID_ZMIANY_STATUSU	NR_REZERWACJI	DATA	NOWY_STATUS
1	61	41	2018-10-18 15:39:56	Z

7.3 TRIGER USUWANIE REZERWACJI

triger zabraniający usunięcia rezerwacji

7.3.1 TRIGER

```
CREATE OR REPLACE TRIGGER USUWANIE_REZERWACJI
BEFORE DELETE
ON REZERWACJE
BEGIN
  RAISE_APPLICATION_ERROR(-20001, 'RECORDS CAN NOT BE DELETED');
END;
```

7.3.2 POMOCNICZA PROCEDURA

```
CREATE OR REPLACE PROCEDURE USUN_REZERWACJE(ID_R NUMBER)
AS
BEGIN
  DECLARE
    REZERWACJA NUMBER;
  BEGIN
    SELECT COUNT(R.NR_REZERWACJI) INTO REZERWACJA FROM REZERWACJE R WHERE R.NR_REZERWACJI = ID_R;

    IF REZERWACJA = 1
    THEN
      DELETE FROM REZERWACJE R WHERE R.NR_REZERWACJI = ID_R;
    END IF;
  END;
END;
```

Po wykonaniu kodu:

```
begin
  USUN_REZERWACJE(6);
end;
```

```
[2018-10-18 17:45:19] [72000][20001] ORA-20001: RECORDS CAN NOT BE DELETED
[2018-10-18 17:45:19] ORA-06512: at "JZIARKO.USUWANIE_REZERWACJI", line 2
[2018-10-18 17:45:19] ORA-04088: error during execution of trigger 'JZIARKO.USUWANIE_REZERWACJI'
[2018-10-18 17:45:19] ORA-06512: at "JZIARKO.USUN_REZERWACJE", line 11
[2018-10-18 17:45:19] ORA-06512: at line 2
```

7.4 UAKTUALNIONE PROCEDURY MODYFIKUJĄCE DANE

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

```
CREATE PROCEDURE ZMIEN_STATUS_REZERWACJI_3(ID_REZERWACJI NUMBER, NOWY_STATUS_ CHAR)
AS
BEGIN
  DECLARE
    ID_R          NUMBER;
    S              CHAR;
    DZISIEJSZA_DATA DATE;
  BEGIN
    SELECT COUNT(R.NR_REZERWACJI) INTO ID_R FROM REZERWACJE R WHERE R.NR_REZERWACJI = ID_REZERWACJI;
    SELECT R.STATUS INTO S FROM REZERWACJE R WHERE R.NR_REZERWACJI = ID_REZERWACJI;
    SELECT CURRENT_DATE INTO DZISIEJSZA_DATA FROM DUAL;
    IF ID_R = 1
    THEN
      IF (S <> 'A') AND NOWY_STATUS_ IN ('N', 'P', 'Z') AND NOWY_STATUS_ <> S
      THEN
        UPDATE REZERWACJE R SET R.STATUS = NOWY_STATUS_ WHERE R.NR_REZERWACJI = ID_REZERWACJI;
      END IF;
    END IF;
  END;
END;
```

8 Polecenie 9

8.1 TRIGER OBSŁUGUJĄCY DODANIE REZERWACJI

Zmiana strategii obsługi redundantnego pola liczba_wolnych_miejsc. realizacja przy pomocy triggerów

8.1.1 DODAJ REZERWACJE 3

```
CREATE PROCEDURE DODAJ_REZERWACJE_3(ID_W NUMBER, ID_O NUMBER)
AS
BEGIN
    DECLARE
        WYCIECZKI NUMBER;
        OSOBA      NUMBER;
    BEGIN
        SELECT COUNT(W.ID_WYCIECZKI) INTO WYCIECZKI FROM WYCIECZKI W WHERE W.ID_WYCIECZKI = ID_W;
        SELECT O.ID_OSOBY INTO OSOBA FROM OSOBY O WHERE O.ID_OSOBY = ID_O;

        IF WYCIECZKI > 0 AND OSOBA > 0
        THEN
            INSERT INTO REZERWACJE (ID_WYCIECZKI, ID_OSOBY, STATUS) VALUES (ID_W, ID_O, 'N');
        END IF;
    END;
END;
```

8.1.2 TRIGER DODAJ REZERWACJE 3

```
CREATE OR REPLACE TRIGGER DODANIE_REZERWACJI_3
AFTER UPDATE
ON REZERWACJE
FOR EACH ROW
DECLARE
    AKTUALNA_DATA DATE;
BEGIN
    SELECT CURRENT_DATE INTO AKTUALNA_DATA FROM DUAL;
    INSERT INTO DZIENNIK_REZERWACJI (NR_REZERWACJI, DATA, NOWY_STATUS)
    VALUES (:OLD.NR_REZERWACJI, AKTUALNA_DATA, :NEW.STATUS);
    BEGIN
        PRZELICZ;
    END;
END;
```

Ten trigger różni się od dodaj_rezerwacje_2 tylko dodaniem procedury przelicz, której poprawne działanie dowiodłem już wcześniej, z tego powodu nie będę zamieszczał zrzutu ekranu.

8.2 TRIGER ZMIANA STATUSU REZERWACJI

trigger obsługujący zmianę statusu Ten sam co poprzednio, ponieważ pole liczba miejsc nie ma związku ze statusem rezerwacji.

```
CREATE OR REPLACE TRIGGER ZMIANA_STATUSU_REZERWACJI
AFTER UPDATE
ON REZERWACJE
FOR EACH ROW
DECLARE
    AKTUALNA_DATA DATE;
BEGIN
    SELECT CURRENT_DATE INTO AKTUALNA_DATA FROM DUAL;
    INSERT INTO DZIENNIK_REZERWACJI (NR_REZERWACJI, DATA, NOWY_STATUS)
```

```
VALUES (:OLD.NR_REZERWACJI, AKTUALNA_DATA, :NEW.STATUS);
END;
```

8.3 ZMIEN LICZBE MIEJSC 3

trigger obsługujący zmianę liczby miejsc na poziomie wycieczki

```
CREATE PROCEDURE ZMIEN_LICZBE_MIEJSC_3(ID_WYCIECZKI_ NUMBER, NOWA_LICZBA_MIEJSC NUMBER)
AS
BEGIN
    DECLARE
        CALKOWITA_LICZBA_MIEJSC NUMBER;
        DOSTEPNE_MIEJSCA_ NUMBER;
    BEGIN
        SELECT W.LICZBA_MIEJSC INTO CALKOWITA_LICZBA_MIEJSC FROM WYCIECZKI W WHERE W.ID_WYCIECZKI = ID_WYCIECZKI_;
        SELECT DOSTEPNE_MIEJSCA(ID_WYCIECZKI_) INTO DOSTEPNE_MIEJSCA_ FROM DUAL;
        IF (NOWA_LICZBA_MIEJSC >= (CALKOWITA_LICZBA_MIEJSC - DOSTEPNE_MIEJSCA_))
        THEN
            UPDATE WYCIECZKI W SET W.LICZBA_MIEJSC = NOWA_LICZBA_MIEJSC WHERE W.ID_WYCIECZKI = ID_WYCIECZKI_;
        END IF;
    END;
END;
```

8.4 TRIGGER ZMIANA LICZBY MIEJSC

```
CREATE OR REPLACE TRIGGER ZMIANA_LICZBY_MIEJSC
before UPDATE
ON WYCIECZKI
FOR EACH ROW
BEGIN
    declare
        VAL number;
    begin
        VAL := :old.LICZBA_MIEJSC - :old.LICZBA_WOLNYCH_MIEJSC;
        :new.LICZBA_WOLNYCH_MIEJSC := :new.LICZBA_MIEJSC - VAL;
    END;
end;
```

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