

Rafał Gawel

Bazy Danych

PL/SQL

1. Tabele	3
2. Wypełniamy tabele przykładowymi danymi	3
3. Tworzenie widoków.	4
4. Tworzenie procedur/funkcji pobierających dane.	7
5. Tworzenie procedur modyfikujących dane.	12
6. Dodajemy tabelę dziennikującą zmiany statusu rezerwacji	16
7. Zmiana struktury bazy danych, w tabeli wycieczki dodajemy redundantne pole liczba_wolnych_miejsc	17
8. Zmiana strategii zapisywania do dziennika rezerwacji. Realizacja przy pomocy triggerów.	25
Oczywiście po wprowadzeniu tej zmiany należy uaktualnić procedury modyfikujące dane. Najlepiej to zrobić tworząc nowe wersje (np. z sufiksem 3)	26
9. Zmiana strategii obsługi redundantnego pola liczba_wolnych_miejsc. Realizacja przy pomocy triggerów	29

1. Tabele

Tabele wstawione zgodnie z instrukcją.

2. Wypełniamy tabele przykładowymi danymi

```
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: 231322, nie dzwonić po 18.00');
INSERT INTO osoby (imie, nazwisko, pesel, kontakt)
VALUES('Jan', 'Kowalski', '12345678', 'tel: 245312, nie dzwonić');
INSERT INTO osoby (imie, nazwisko, pesel, kontakt)
VALUES('Stansław', 'Nowak', '12345678', 'tel: 762312,');
INSERT INTO osoby (imie, nazwisko, pesel, kontakt)
VALUES('Janusz', 'Kowalski', '12345678', 'tel: 982312, stacjonarny');

INSERT INTO wycieczki (nazwa, kraj, data, opis, liczba_miejsc)
VALUES ('Wycieczka do Paryża', 'Francja', TO_DATE('2016-01-01', 'YYYY-MM-DD'),
'Ciekawa wycieczka ...',3);
INSERT INTO wycieczki (nazwa, kraj, data, opis, liczba_miejsc)
VALUES ('Piękny Kraków', 'Polska', TO_DATE('2017-02-03', 'YYYY-MM-DD'),'Najciekawa
wycieczka ...',2);
INSERT INTO wycieczki (nazwa, kraj, data, opis, liczba_miejsc)
VALUES ('Wieliczka', 'Polska', TO_DATE('2017-03-03', 'YYYY-MM-DD'),'Zadziwiająca
kopalnia ...',2);
INSERT INTO wycieczki (nazwa, kraj, data, opis, liczba_miejsc)
VALUES ('Monachium', 'Niemcy', '2020-03-03', 'Zadziwiające miasto ...',5);
INSERT INTO wycieczki (nazwa, kraj, data, opis, liczba_miejsc)
VALUES ('Gniezno', 'Polska', TO_DATE('2020-05-02', 'YYYY-MM-DD'), 'Stere miasto
...',2);
INSERT INTO wycieczki (nazwa, kraj, data, opis, liczba_miejsc)
VALUES ('Monachium', 'Niemcy', TO_DATE('2020-03-03', 'YYYY-MM-DD')
, 'Zadziwiające miasto ...',5);
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 (21,21,'N');
INSERT INTO rezerwacje(id_wycieczki, id_osoby, status)
VALUES (21,22,'P');
INSERT INTO rezerwacje(id_wycieczki, id_osoby, status)
VALUES (21,41,'A');
```

3. Tworzenie widoków.

Należy przygotować kilka widoków ułatwiających dostęp do danych

- a) wycieczki_osoby(kraj, data, nazwa_wycieczki, imie, nazwisko, status_rezerwacji)

```
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
```

	ID_WYCIECZKI	NAZWA	KRAJ	DATA	IMIE	NAZWISKO	STATUS
1	2	Wycieczka do Paryza	Francja	2016-01-01 00:00:00	Adam	Kowalski	N
2	2	Wycieczka do Paryza	Francja	2016-01-01 00:00:00	Jan	Nowak	P
3	3	Piękny Kraków	Polska	2017-02-03 00:00:00	Jan	Nowak	N
4	21	Monachium	Niemcy	2020-03-03 00:00:00	Jan	Nowak	P
5	21	Monachium	Niemcy	2020-03-03 00:00:00	Jan	Kowalski	N
6	21	Monachium	Niemcy	2020-03-03 00:00:00	Stansław	Nowak	P
7	21	Monachium	Niemcy	2020-03-03 00:00:00	Janusz	Kowalski	A

- b) wycieczki_osoby_potwierdzone(kraj, data, nazwa_wycieczki, imie, nazwisko, status_rezerwacji)

```
CREATE VIEW WYCIECZKI_OSOBY_POTWIERDZONE
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
WHERE r.STATUS = 'P' OR r.STATUS = 'Z'
```

	ID_WYCIECZKI	NAZWA	KRAJ	DATA	IMIE	NAZWISKO	STATUS
1	2	Wycieczka do Paryża	Francja	2016-01-01 00:00:00	Jan	Nowak	P
2	21	Monachium	Niemcy	2020-03-03 00:00:00	Jan	Nowak	P
3	21	Monachium	Niemcy	2020-03-03 00:00:00	Stasiław	Nowak	P

c) wycieczki_przyszle(kraj, data, nazwa_wycieczki, imie, nazwisko, status rezerwacji)

```
CREATE VIEW WYCIECZKI_PRZYSZLE
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
WHERE w.DATA > CURRENT_DATE
```

	ID_WYCIECZKI	NAZWA	KRAJ	DATA	IMIE	NAZWISKO	STATUS
1	21	Monachium	Niemcy	2020-03-03 00:00:00	Jan	Nowak	P
2	21	Monachium	Niemcy	2020-03-03 00:00:00	Jan	Kowalski	N
3	21	Monachium	Niemcy	2020-03-03 00:00:00	Stasiław	Nowak	P
4	21	Monachium	Niemcy	2020-03-03 00:00:00	Janusz	Kowalski	A

d) wycieczki_miejsca(kraj, data, nazwa_wycieczki, liczba_miejsc, liczba_wolnych miejsc)

```
CREATE VIEW WYCIECZKI_MIEJSCA
AS
SELECT
    w.ID_WYCIECZKI,
    w.NAZWA,
    w.KRAJ,
    w.DATA,
    w.LICZBA_MIEJSC,
    w.LICZBA_MIEJSC - (select count (*) from REZERWACJE where
                        REZERWACJE.ID_WYCIECZKI = W.ID_WYCIECZKI)
                        as liczba_wolnych_miejsc
FROM WYCIECZKI w
    LEFT JOIN (SELECT * FROM REZERWACJE WHERE STATUS <> 'A') r
    ON r.ID_WYCIECZKI = w.ID_WYCIECZKI
GROUP BY w.ID_WYCIECZKI, w.NAZWA, w.KRAJ, w.DATA,
    w.LICZBA_MIEJSC
```

	ID_WYCIECZKI	NAZWA	KRAJ	DATA	LICZBA_MIEJSC	LICZBA_WOLNYCH_MIEJSC
1	2	Wycieczka do Paryża	Francja	2016-01-01 00:00:00	3	1
2	3	Piękny Kraków	Polska	2017-02-03 00:00:00	2	1
3	4	Wieliczka	Polska	2017-03-03 00:00:00	2	2
4	21	Monachium	Niemcy	2020-03-03 00:00:00	5	1
5	42	Gniezno	Polska	2020-05-02 00:00:00	2	2

e) dostępne_wycieczki(kraj, data, nazwa_wycieczki, liczba_miejsc, liczba_wolnych miejsc)

```

CREATE VIEW DOSTEPNE_WYCIECZKI
AS
SELECT
    w.ID_WYCIECZKI,
    w.NAZWA,
    w.KRAJ,
    w.DATA,
    w.LICZBA_MIEJSC,
    w.LICZBA_WOLNYCH_MIEJSC
FROM WYCIECZKI_MIEJSCA w
WHERE LICZBA_WOLNYCH_MIEJSC > 0
    AND w.DATA > CURRENT_DATE

```

	ID_WYCIECZKI	NAZWA	KRAJ	DATA	LICZBA_MIEJSC	LICZBA_WOLNYCH_MIEJSC
1	21	Monachium	Niemcy	2020-03-03 00:00:00	5	1
2	42	Gniezno	Polska	2020-05-02 00:00:00	2	2

- f) rezerwacje_do_anulowania - lista niepotwierdzonych rezerwacji które powinny zostać anulowane (rezerwacje przygotowywane są do anulowania na tydzień przed wyjazdem)

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

	IMIE	NAZWISKO	STATUS	NAZWA	DATA
1	Jan	Nowak	N	Piękny Kraków	2017-02-03 00:00:00
2	Adam	Kowalski	N	Wycieczka do Paryża	2016-01-01 00:00:00

4. Tworzenie procedur/funkcji pobierających dane.

Podobnie jak w poprzednim przykładzie należy przygotować kilka procedur ułatwiających dostęp do danych

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

```
CREATE OR REPLACE TYPE wycieczki_osoby_row AS OBJECT (  
    KRAJ          VARCHAR(50),  
    "DATA"        DATE,  
    NAZWA         VARCHAR(100),  
    IMIE          VARCHAR2(50),  
    NAZWISKO      VARCHAR2(50),  
    STATUS        CHAR(1)  
);
```

```
CREATE OR REPLACE TYPE wycieczki_osoby_table IS TABLE OF  
wycieczki_osoby_row;
```

```
CREATE OR REPLACE  
FUNCTION uczestnicy_wycieczki(id WYCIECZKI.ID_WYCIECZKI%type)  
RETURN wycieczki_osoby_table AS result wycieczki_osoby_table;  
istnieje          INT;  
BEGIN  
    SELECT COUNT(*) INTO istnieje FROM WYCIECZKI WHERE  
WYCIECZKI.ID_WYCIECZKI = id;  
  
    IF istnieje = 0 THEN  
        raise_application_error(-20001, 'Wycieczka o podanym id nie istnieje');  
    END IF;  
  
    SELECT wycieczki_osoby_row(w.KRAJ, w.DATA, w.NAZWA, o.IMIE,  
                              o.NAZWISKO, r.STATUS)  
        BULK COLLECT INTO result  
    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;  
    return result;  
END;
```


Przykład użycia i rezultat:

```
select * from table(uczestnicy_wycieczki(2))
```

	KRAJ	DATA	NAZWA	IMIE	NAZWISKO	STATUS
1	Francja	2016-01-01 00:00:00	Wycieczka do Paryża	Adam	Kowalski	N
2	Francja	2016-01-01 00:00:00	Wycieczka do Paryża	Jan	Nowak	P

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

CREATE OR REPLACE

```
FUNCTION rezerwacje_osoby(id OSOBY.ID_OSOBY%type)
RETURN wycieczki_osoby_table AS result wycieczki_osoby_table;
istnieje INT;
BEGIN
    SELECT COUNT(*) INTO istnieje FROM OSOBY WHERE OSOBY.ID_OSOBY = id;

    IF istnieje = 0 THEN
        raise_application_error(-20002, 'Osoba o podanym id nie istnieje');
    END IF;

    SELECT wycieczki_osoby_row(w.KRAJ, w.DATA, w.NAZWA, o.IMIE,
        o.NAZWISKO, r.STATUS)
        BULK COLLECT INTO result
    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;
    RETURN result;
END;
```

Przykład użycia i rezultat:

```
select * from table(rezerwacje_osoby(3))
```

	KRAJ	DATA	NAZWA	IMIE	NAZWISKO	STATUS
1	Francja	2016-01-01 00:00:00	Wycieczka do Paryża	Jan	Nowak	P
2	Polska	2017-02-03 00:00:00	Piękny Kraków	Jan	Nowak	N
3	Niemcy	2020-03-03 00:00:00	Monachium	Jan	Nowak	P

c) przyszle_rezerwacje_osoby(id_osoby)

CREATE OR REPLACE

```
FUNCTION przyszle_rezerwacje_osoby(id OSOBY.ID_OSOBY%type)
  RETURN wycieczki_osoby_table AS result wycieczki_osoby_table;
istnieje INT;
BEGIN
  SELECT COUNT(*) INTO istnieje FROM OSOBY WHERE OSOBY.ID_OSOBY = id;

  IF istnieje = 0 THEN
    raise_application_error(-20002, 'Osoba o podanym id nie istnieje');
  END IF;

  SELECT wycieczki_osoby_row(w.KRAJ, w.DATA, w.NAZWA, o.IMIE,
    o.NAZWISKO, r.STATUS)
    BULK COLLECT INTO result
  FROM WYCIEZKI w
    JOIN REZERWACJE r ON w.ID_WYCIEZKI = r.ID_WYCIEZKI
    JOIN OSOBY o ON r.ID_OSOBY = o.ID_OSOBY
  WHERE o.ID_OSOBY = id AND w.DATA > CURRENT_DATE;
  RETURN result;
END;
```

Przykład użycia i rezultat:

```
select * from table(przyszle_rezerwacje_osoby(2))
```

	KRAJ	DATA	NAZWA	IMIE	NAZWISKO	STATUS
1	Niemcy	2020-03-03 00:00:00	Monachium	Jan	Nowak	P

d) `dostepne_wycieczki(kraj, data_od, data_do)`

```
CREATE OR REPLACE TYPE wycieczki_row AS OBJECT (  
    ID_WYCIECZKI INT,  
    NAZWA        VARCHAR2(100),  
    KRAJ         VARCHAR2(50),  
    "DATA"       DATE,  
    OPIS         VARCHAR2(200),  
    LICZBA_MIEJSC INT  
);
```

```
CREATE OR REPLACE TYPE wycieczki_table IS TABLE OF wycieczki_row;
```

```
CREATE OR REPLACE
```

```
FUNCTION dostepne_wycieczki_f(kraj WYCIECZKI.KRAJ%TYPE, data_od DATE, data_do  
DATE)
```

```
RETURN wycieczki_table AS result wycieczki_table;
```

```
BEGIN
```

```
    IF data_do < data_od THEN
```

```
        raise_application_error(-20003, 'Nieprawidłowa data');
```

```
    END IF;
```

```
    SELECT wycieczki_row(w.ID_WYCIECZKI, w.NAZWA, w.KRAJ, w.DATA, w.OPIS,  
        w.LICZBA_MIEJSC)
```

```
        BULK COLLECT INTO result
```

```
    FROM WYCIECZKI w
```

```
    WHERE w.KRAJ = dostepne_wycieczki_f.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 result;
```

```
END;
```

Przykład użycia i rezultat:

```
select * FROM table(dostepne_wycieczki_f2('Polska', TO_DATE('2016-11-30',  
'YYYY-MM-DD'), TO_DATE('2021-01-01', 'YYYY-MM-DD')));
```

	ID_WYCIECZKI	NAZWA	KRAJ	DATA	OPIS	LICZBA_MIEJSC
1	3	Piękny Kraków	Polska	2017-02-03 00:00:00	Najciekawa wycieczka ...	2
2	4	Wieliczka	Polska	2017-03-03 00:00:00	Zadziwiająca kopalnia ...	2
3	42	Gniezno	Polska	2020-05-02 00:00:00	Stere miasto ...	2

5. Tworzenie procedur modyfikujących dane.

Należy przygotować zestaw procedur pozwalających na modyfikację danych oraz kontrolę poprawności ich wprowadzania

a) `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_wycieczki WYCIECZKI.ID_WYCIECZKI%TYPE, id_osoby  
OSOBY.ID_OSOBY%TYPE) AS
```

```
test integer;
```

```
BEGIN
```

```
SELECT COUNT(*) INTO test
```

```
FROM WYCIECZKI w
```

```
WHERE w.ID_WYCIECZKI = dodaj_rezerwacje.id_wycieczki;
```

```
IF test = 0 THEN
```

```
raise_application_error(-20004, 'Wycieczka o podanym id nie istnieje');
```

```
END IF;
```

```
SELECT COUNT(*) INTO test
```

```
FROM REZERWACJE r
```

```
WHERE r.ID_WYCIECZKI = dodaj_rezerwacje.id_wycieczki
```

```
AND r.ID_OSOBY = dodaj_rezerwacje.id_osoby;
```

```
IF test > 0 THEN
```

```
raise_application_error(-20005, 'Rezerwacja już istnieje');
```

```
END IF;
```

```
SELECT COUNT(*) INTO test
```

```
FROM OSOBY
```

```
WHERE OSOBY.ID_OSOBY = dodaj_rezerwacje.id_osoby;
```

```
IF test = 0 THEN
```

```
raise_application_error(-20002, 'Osoba o podanym id nie istnieje');
```

```
END IF;
```

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

```
VALUES (dodaj_rezerwacje.id_wycieczki, dodaj_rezerwacje.id_osoby, 'N');
```

```
END;
```

Przykład użycia i rezultat:

begin

dodaj_rezerwacje(42, 21);

end;

	NR_REZERWACJI	ID_WYCIECZKI	ID_OSOBY	STATUS
1	64	21	21	P
2	65	21	22	P
3	66	21	41	A
4	81	42	21	N
5	2	2	1	N
6	3	2	2	P
7	29	3	2	N
8	41	21	2	P

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

CREATE OR REPLACE PROCEDURE

zmiana_statusu_rezerwacji(id_rezerwacji REZERWACJE.NR_REZERWACJI%TYPE,
nowy_status REZERWACJE.STATUS%TYPE) AS

stary_status REZERWACJE.STATUS%TYPE;

test integer;

BEGIN

SELECT COUNT(*) **INTO** test

FROM WYCIECZKI_PRZYSZLE wp

JOIN REZERWACJE r **ON** r.ID_WYCIECZKI = wp.ID_WYCIECZKI

WHERE r.NR_REZERWACJI = id_rezerwacji;

IF test = 0

THEN

raise_application_error(-20010,

'Nie można zmienić statusu rezerwacji dla wycieczki która już się odbyła');

END IF;

SELECT status **INTO** stary_status

FROM REZERWACJE

WHERE NR_REZERWACJI = id_rezerwacji;

CASE

WHEN stary_status **IS NULL** **THEN**

raise_application_error(-20011, 'Rezerwacja nie istnieje');

WHEN nowy_status = 'N' **THEN**

raise_application_error(-20012,

'Istniejąca rezerwacja nie może stać się nowa');

```
WHEN stary_status = 'A' THEN
  SELECT COUNT(*) INTO test
  FROM DOSTEPNE_WYCIECZKI dwv
    JOIN REZERWACJE r ON r.ID_WYCIECZKI = dwv.ID_WYCIECZKI
  WHERE r.NR_REZERWACJI = id_rezerwacji;

  IF test = 0 THEN
    raise_application_error(-20013,
      'Brak miejsc dla przywrócenia anulowanej rezerwacji');
  END IF;
ELSE null;
END CASE;

UPDATE REZERWACJE
SET STATUS = nowy_status
WHERE NR_REZERWACJI = id_rezerwacji;

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

END;
```

Przykład użycia i rezultat:

```
begin
  zmiana_statusu_rezerwacji(41, 'P');
end;
```

	NR_REZERWACJI	ID_WYCIECZKI	ID_OSOBY	STATUS
1	64	21	21	P
2	65	21	22	P
3	66	21	41	A
4	81	42	21	P
5	2	2	1	N
6	3	2	2	P
7	29	3	2	N
8	41	21	2	P

Po dodaniu tabeli dziennikującej (z następnego podpunktu) w niej też zmiana zostaje odnotowana.

	ID	ID_REZERWACJI	DATA	STATUS
1	21	64	2019-10-19 21:14:09	P
2	42	81	2019-10-22 19:05:32	P
3	1	41	2019-10-19 13:57:25	P

c) 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.ID_WYCIECZKI%TYPE, nowe_miejsca
WYCIECZKI.LICZBA_MIEJSC%TYPE) AS
test integer;
BEGIN
    SELECT LICZBA_MIEJSC - LICZBA_WOLNYCH_MIEJSC INTO test
    FROM WYCIECZKI_MIEJSCA w
    WHERE w.ID_WYCIECZKI = id;

    IF nowe_miejsca < 0 OR test > nowe_miejsca THEN
        raise_application_error(-20014, 'Podana liczba miejsc jest za mała');
    END IF;

    UPDATE WYCIECZKI
    SET LICZBA_MIEJSC = nowe_miejsca
    WHERE ID_WYCIECZKI = id;
END;
```

Przykład użycia i rezultat:

begin

 zmien_liczbe_miejsc(42, 3);

end;

ID_WYCIECZKI	NAZWA	KRAJ	DATA	OPIS	LICZBA_MIEJSC	LICZBA_WOLNYCH_MIEJSC
1	2 Wycieczka do Paryża	Francja	2016-01-01 00:00:00	Ciekawa wycieczka ...	3	1
2	3 Piękny Kraków	Polska	2017-02-03 00:00:00	Najciekawsza wycieczka ...	2	1
3	4 Wieliczka	Polska	2017-03-03 00:00:00	Zadziwiająca kopalnia ...	2	2
4	21 Monachium	Niemcy	2020-03-03 00:00:00	Zadziwiające miasto ...	5	2
5	42 Gniezno	Polska	2020-05-02 00:00:00	Stere miasto ...	3	2

6. Dodajemy tabelę dziennikującą zmiany statusu rezerwacji

rezerwacje_log(id, id_rezerwacje, data, status)

Tworzenie tabeli dziennikującej

```
CREATE TABLE REZERWACJE_LOG (  
  ID INT GENERATED ALWAYS AS IDENTITY NOT NULL,  
  ID_REZERWACJI INT,  
  DATA DATE default sysdate not null,  
  STATUS CHAR(1),  
  CONSTRAINT REZERWACJE_LOG_PK PRIMARY KEY (  
    ID  
  ) ENABLE  
);
```

Dodanie do tabeli rezerwacje_log relacji z tabela rezerwacje

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

Dodanie check'a na statusie w tabeli rezerwacje_log

```
ALTER TABLE REZERWACJE_LOG  
ADD CONSTRAINT REZERWACJE_LOG_CHK1 CHECK  
(STATUS in ('N','P','Z','A'))  
ENABLE ;
```


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

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

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

```
CREATE OR REPLACE PROCEDURE AKTUALIZACJA_WOLNYCH_MIEJSC  
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 w.LICZBA_WOLNYCH_MIEJSC is null;  
END;
```

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

```
CREATE OR REPLACE VIEW dostepne_wycieczki2  
AS  
SELECT  
    w.ID_WYCIECZKI,  
    w.NAZWA,  
    w.KRAJ,  
    w.DATA,  
    w.LICZBA_MIEJSC,  
    w.LICZBA_WOLNYCH_MIEJSC  
FROM WYCIECZKI w  
WHERE liczba_wolnych_miejsc > 0  
    AND w.DATA > CURRENT_DATE;
```

CREATE OR REPLACE VIEW wycieczki_miejsca2

AS

```
SELECT w.ID_WYCIECZKI,  
       w.NAZWA,  
       w.KRAJ,  
       w.DATA,  
       w.LICZBA_MIEJSC,  
       w.LICZBA_WOLNYCH_MIEJSC  
FROM wycieczki w;
```

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

CREATE OR REPLACE

FUNCTION *dostepne_wycieczki_f2*(kraj WYCIECZKI.KRAJ%TYPE, data_od DATE, data_do DATE)

RETURN wycieczki_table **AS** result wycieczki_table;

BEGIN

IF data_do < data_od **THEN**

raise_application_error(-20003, 'Nieprawidłowa data');

END IF;

SELECT wycieczki_row(w.ID_WYCIECZKI, w.NAZWA, w.KRAJ, w.DATA, w.OPIS, w.LICZBA_MIEJSC)

BULK COLLECT INTO result

FROM WYCIECZKI w

WHERE w.KRAJ = *dostepne_wycieczki_f2*.kraj

AND w.DATA >= data_od

AND w.DATA <= data_do

AND w.LICZBA_WOLNYCH_MIEJSC > 0;

RETURN result;

END;

select * **FROM** table(*dostepne_wycieczki_f2*('Francja', TO_DATE('2015-01-01', 'YYYY-MM-DD'), TO_DATE('2020-01-01', 'YYYY-MM-DD')));

	ID_WYCIECZKI	NAZWA	KRAJ	DATA	OPIS	LICZBA_MIEJSC
1	2	Wycieczka do Paryza	Francja	2016-01-01 00:00:00	Ciekawa wycieczka ...	3

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)

CREATE OR REPLACE PROCEDURE

dodaj_rezerwacje2(id_wycieczki WYCIECZKI.ID_WYCIECZKI%TYPE, id_osoby OSOBY.ID_OSOBY%TYPE) AS

test integer;

BEGIN

SELECT COUNT(*) INTO test

FROM WYCIECZKI w

WHERE w.ID_WYCIECZKI = *dodaj_rezerwacje2*.id_wycieczki;

IF test = 0 THEN

raise_application_error(-20004, 'Wycieczka o podanym id nie istnieje');

END IF;

SELECT COUNT(*) INTO test

FROM REZERWACJE r

WHERE r.ID_WYCIECZKI = *dodaj_rezerwacje2*.id_wycieczki

AND r.ID_OSOBY = *dodaj_rezerwacje2*.id_osoby;

IF test > 0 THEN

raise_application_error(-20005, 'Rezerwacja już istnieje');

END IF;

SELECT COUNT(*) INTO test

FROM OSOBY

WHERE OSOBY.ID_OSOBY = *dodaj_rezerwacje2*.id_osoby;

IF test = 0 THEN

raise_application_error(-20002, 'Osoba o podanym id nie istnieje');

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;

CREATE OR REPLACE PROCEDURE

```
zmiana_statusu_rezerwacji2(id_rezerwacji REZERWACJE.NR_REZERWACJI%TYPE,  
nowy_status REZERWACJE.STATUS%TYPE) AS  
stary_status REZERWACJE.STATUS%TYPE;  
test integer;  
tmp integer;  
BEGIN  
    SELECT COUNT(*) INTO test  
    FROM WYCIEZKI_PRZYSZLE wp  
    JOIN REZERWACJE r ON r.ID_WYCIEZKI = wp.ID_WYCIEZKI  
    WHERE r.NR_REZERWACJI = id_rezerwacji;  
    IF test = 0  
    THEN  
        raise_application_error(-20010,  
                                'Nie można zmienić statusu rezerwacji dla wycieczki która już się  
                                odbyła');  
    END IF;  
  
    SELECT status INTO stary_status  
    FROM REZERWACJE  
    WHERE NR_REZERWACJI = id_rezerwacji;  
  
    CASE  
        WHEN stary_status IS NULL THEN  
            raise_application_error(-20011, 'Rezerwacja nie istnieje');  
  
        WHEN nowy_status = 'N' THEN  
            raise_application_error(-20012,  
                                'Istniejąca rezerwacja nie może stać się nowa');  
  
        WHEN stary_status = 'A' THEN  
            SELECT COUNT(*) INTO test  
            FROM DOSTEPNE_WYCIEZKI dw  
            JOIN REZERWACJE r ON r.ID_WYCIEZKI = dw.ID_WYCIEZKI  
            WHERE r.NR_REZERWACJI = id_rezerwacji;  
  
            IF test = 0 THEN  
                raise_application_error(-20013,  
                                        'Brak miejsc dla przywrócenia anulowanej rezerwacji');  
            ELSE  
                tmp := -1;  
            END IF;  
        ELSE  
            IF nowy_status = 'A' THEN  
                tmp := 1;  
            ELSE
```

```
    tmp := 0;  
end if;  
END CASE;
```

```
UPDATE REZERWACJE  
SET STATUS = nowy_status  
WHERE NR_REZERWACJI = id_rezerwacji;
```

```
UPDATE WYCIECZKI w  
SET LICZBA_WOLNYCH_MIEJSC = LICZBA_WOLNYCH_MIEJSC + tmp  
WHERE w.ID_WYCIECZKI = (SELECT ID_WYCIECZKI  
    FROM REZERWACJE r  
    WHERE r.NR_REZERWACJI = id_rezerwacji);
```

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

```
END;
```

```

CREATE OR REPLACE PROCEDURE zmien_liczbe_miejsc2(id
WYCIECZKI.ID_WYCIECZKI%TYPE, nowe_miejsca
WYCIECZKI.LICZBA_MIEJSC%TYPE) AS
test integer;
BEGIN
    SELECT LICZBA_MIEJSC - LICZBA_WOLNYCH_MIEJSC INTO test
    FROM WYCIECZKI w
    WHERE w.ID_WYCIECZKI = id;

    IF nowe_miejsca < 0 OR test > nowe_miejsca THEN
        raise_application_error(-20014, 'Podana liczba miejsc jest za mała');
    END IF;

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

```

```

CREATE OR REPLACE PROCEDURE
zmiana_statusu_rezerwacji3(id_rezerwacji REZERWACJE.NR_REZERWACJI%TYPE,
nowy_status REZERWACJE.STATUS%TYPE) AS
stary_status REZERWACJE.STATUS%TYPE;
test integer;
BEGIN
    SELECT COUNT(*) INTO test
    FROM WYCIECZKI_PRZYSZLE wp
    JOIN REZERWACJE r ON r.ID_WYCIECZKI = wp.ID_WYCIECZKI
    WHERE r.NR_REZERWACJI = id_rezerwacji;
    IF test = 0
    THEN
        raise_application_error(-20010,
            'Nie można zmienić statusu rezerwacji dla wycieczki która już się
odbyła');
    END IF;

    SELECT status INTO stary_status
    FROM REZERWACJE
    WHERE NR_REZERWACJI = id_rezerwacji;

    CASE
    WHEN stary_status IS NULL THEN
        raise_application_error(-20011, 'Rezerwacja nie istnieje');

```

```

WHEN nowy_status = 'N' THEN
    raise_application_error(-20012,
        'Istniejąca rezerwacja nie może stać się nowa');

WHEN stary_status = 'A' THEN
    SELECT COUNT(*) INTO test
    FROM DOSTEPNE_WYCIECZKI dwv
        JOIN REZERWACJE r ON r.ID_WYCIECZKI = dwv.ID_WYCIECZKI
    WHERE r.NR_REZERWACJI = id_rezerwacji;

    IF test = 0 THEN
        raise_application_error(-20013,
            'Brak miejsc dla przywrócenia anulowanej rezerwacji');
    END IF;
    ELSE null;
END CASE;

UPDATE REZERWACJE
SET STATUS = nowy_status
WHERE NR_REZERWACJI = id_rezerwacji;
END;

```

Dodanie jednego prostego pola do tabeli powoduje dużo zmian nawet w nieskomplikowanej bazie danych

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

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

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

```
CREATE OR REPLACE TRIGGER triger_zabraniajacy_usuniecia_rezerwacji
BEFORE DELETE ON REZERWACJE
FOR EACH ROW
BEGIN
    raise_application_error(-20020, 'Rezerwacja nie moze zostać usunięta');
END;
```


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

CREATE OR REPLACE PROCEDURE

dodaj_rezerwacje3(id_wycieczki WYCIECZKI.ID_WYCIECZKI%TYPE, id_osoby OSOBY.ID_OSOBY%TYPE) **AS**

test integer;

BEGIN

SELECT COUNT(*) INTO test

FROM WYCIECZKI **w**

WHERE w.ID_WYCIECZKI = *dodaj_rezerwacje3*.id_wycieczki;

IF test = 0 THEN

raise_application_error(-20004, 'Wycieczka o podanym id nie istnieje');

END IF;

SELECT COUNT(*) INTO test

FROM REZERWACJE **r**

WHERE r.ID_WYCIECZKI = *dodaj_rezerwacje3*.id_wycieczki
AND r.ID_OSOBY = *dodaj_rezerwacje3*.id_osoby;

IF test > 0 THEN

raise_application_error(-20005, 'Rezerwacja już istnieje');

END IF;

SELECT COUNT(*) INTO test

FROM OSOBY

WHERE OSOBY.ID_OSOBY = *dodaj_rezerwacje3*.id_osoby;

IF test = 0 THEN

raise_application_error(-20002, 'Osoba o podanym id nie istnieje');

END IF;

INSERT INTO REZERWACJE (ID_WYCIECZKI, ID_OSOBY, STATUS)

VALUES (*dodaj_rezerwacje3*.id_wycieczki, *dodaj_rezerwacje3*.id_osoby, 'N');

UPDATE WYCIECZKI

SET LICZBA_WOLNYCH_MIEJSC = LICZBA_WOLNYCH_MIEJSC - 1

WHERE ID_WYCIECZKI = *dodaj_rezerwacje3*.id_wycieczki;

END;

CREATE PROCEDURE

```
zmiana_statusu_rezerwacji2(id_rezerwacji REZERWACJE.NR_REZERWACJI%TYPE,  
nowy_status REZERWACJE.STATUS%TYPE) AS  
stary_status REZERWACJE.STATUS%TYPE;  
test integer;  
tmp integer;  
BEGIN  
    SELECT COUNT(*) INTO test  
    FROM WYCIECZKI_PRZYSZLE wp  
    JOIN REZERWACJE r ON r.ID_WYCIECZKI = wp.ID_WYCIECZKI  
    WHERE r.NR_REZERWACJI = id_rezerwacji;  
    IF test = 0  
    THEN  
        raise_application_error(-20010,  
                                'Nie można zmienić statusu rezerwacji dla wycieczki która już się  
odbyła');  
    END IF;  
  
    SELECT status INTO stary_status  
    FROM REZERWACJE  
    WHERE NR_REZERWACJI = id_rezerwacji;  
  
    CASE  
        WHEN stary_status IS NULL THEN  
            raise_application_error(-20011, 'Rezerwacja nie istnieje');  
  
        WHEN nowy_status = 'N' THEN  
            raise_application_error(-20012,  
                                    'Istniejąca rezerwacja nie może stać się nowa');  
  
        WHEN stary_status = 'A' THEN  
            SELECT COUNT(*) INTO test  
            FROM DOSTEPNE_WYCIECZKI dw  
            JOIN REZERWACJE r ON r.ID_WYCIECZKI = dw.ID_WYCIECZKI  
            WHERE r.NR_REZERWACJI = id_rezerwacji;  
  
            IF test = 0 THEN  
                raise_application_error(-20013,  
                                        'Brak miejsc dla przywrócenia anulowanej rezerwacji');  
            ELSE  
                tmp := -1;  
            END IF;  
        ELSE  
            IF nowy_status = 'A' THEN  
                tmp := 1;  
            ELSE
```

```
    tmp := 0;  
    end if;  
END CASE;
```

```
UPDATE REZERWACJE  
SET STATUS = nowy_status  
WHERE NR_REZERWACJI = id_rezerwacji;
```

```
UPDATE WYCIEZKI w  
SET LICZBA_WOLNYCH_MIEJSC = LICZBA_WOLNYCH_MIEJSC + tmp  
WHERE w.ID_WYCIEZKI = (SELECT ID_WYCIEZKI  
    FROM REZERWACJE r  
    WHERE r.NR_REZERWACJI = id_rezerwacji);
```

```
END;
```

9. Zmiana strategii obsługi redundantnego pola liczba_wolnych_miejsc. Realizacja przy pomocy triggerów

```
CREATE OR REPLACE TRIGGER triger_obslugujacy_dodanie_rezerwacji
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;
```

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

    UPDATE WYCIECZKI w
    SET LICZBA_WOLNYCH_MIEJSC = tmp + LICZBA_WOLNYCH_MIEJSC
    WHERE w.ID_WYCIECZKI = :NEW.ID_WYCIECZKI;
END;
```

```

CREATE OR REPLACE TRIGGER triger_obsługujacy_zmiane_liczby_miejsc_wycieczki
BEFORE UPDATE OF liczba_miejsc ON wycieczki
FOR EACH ROW
BEGIN
    :NEW.liczba_wolnych_miejsc := :NEW.LICZBA_MIEJSC - (:OLD.LICZBA_MIEJSC -
:OLD.liczba_wolnych_miejsc);
END;

```

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

```

CREATE OR REPLACE PROCEDURE
    dodaj_rezerwacje3(id_wycieczki WYCIECZKI.ID_WYCIECZKI%TYPE, id_osoby
OSOBY.ID_OSOBY%TYPE) AS
    test integer;
BEGIN
    SELECT COUNT(*) INTO test
    FROM WYCIECZKI w
    WHERE w.ID_WYCIECZKI = dodaj_rezerwacje3.id_wycieczki;

    IF test = 0 THEN
        raise_application_error(-20004, 'Wycieczka o podanym id nie istnieje');
    END IF;

    SELECT COUNT(*) INTO test
    FROM REZERWACJE r
    WHERE r.ID_WYCIECZKI = dodaj_rezerwacje3.id_wycieczki
        AND r.ID_OSOBY = dodaj_rezerwacje3.id_osoby;

    IF test > 0 THEN
        raise_application_error(-20005, 'Rezerwacja już istnieje');
    END IF;

    SELECT COUNT(*) INTO test
    FROM OSOBY
    WHERE OSOBY.ID_OSOBY = dodaj_rezerwacje3.id_osoby;

    IF test = 0 THEN
        raise_application_error(-20002, 'Osoba o podanym id nie istnieje');
    END IF;

    INSERT INTO REZERWACJE (ID_WYCIECZKI, ID_OSOBY, STATUS)
    VALUES (dodaj_rezerwacje3.id_wycieczki, dodaj_rezerwacje3.id_osoby, 'N');

END;

```

CREATE OR REPLACE PROCEDURE

```
zmiana_statusu_rezerwacji3(id_rezerwacji REZERWACJE.NR_REZERWACJI%TYPE,  
nowy_status REZERWACJE.STATUS%TYPE) AS  
stary_status REZERWACJE.STATUS%TYPE;  
test integer;  
BEGIN  
    SELECT COUNT(*) INTO test  
    FROM WYCIECZKI_PRZYSZLE wp  
    JOIN REZERWACJE r ON r.ID_WYCIECZKI = wp.ID_WYCIECZKI  
    WHERE r.NR_REZERWACJI = id_rezerwacji;  
    IF test = 0  
    THEN  
        raise_application_error(-20010,  
                                'Nie można zmienić statusu rezerwacji dla wycieczki która już się  
odbyła');  
    END IF;  
  
    SELECT status INTO stary_status  
    FROM REZERWACJE  
    WHERE NR_REZERWACJI = id_rezerwacji;  
  
    CASE  
        WHEN stary_status IS NULL THEN  
            raise_application_error(-20011, 'Rezerwacja nie istnieje');  
  
        WHEN nowy_status = 'N' THEN  
            raise_application_error(-20012,  
                                    'Istniejąca rezerwacja nie może stać się nowa');  
  
        WHEN stary_status = 'A' THEN  
            SELECT COUNT(*) INTO test  
            FROM DOSTEPNE_WYCIECZKI dwv  
            JOIN REZERWACJE r ON r.ID_WYCIECZKI = dwv.ID_WYCIECZKI  
            WHERE r.NR_REZERWACJI = id_rezerwacji;  
  
            IF test = 0 THEN  
                raise_application_error(-20013,  
                                        'Brak miejsc dla przywrócenia anulowanej rezerwacji');  
            END IF;  
        ELSE null;  
    END CASE;  
  
    UPDATE REZERWACJE  
    SET STATUS = nowy_status  
    WHERE NR_REZERWACJI = id_rezerwacji;  
END;
```

```

CREATE OR REPLACE PROCEDURE zmien_liczbe_miejsc3(id
WYCIECZKI.ID_WYCIECZKI%TYPE, nowe_miejsca
WYCIECZKI.LICZBA_MIEJSC%TYPE) AS
test integer;
BEGIN
    SELECT LICZBA_MIEJSC - LICZBA_WOLNYCH_MIEJSC INTO test
    FROM WYCIECZKI w
    WHERE w.ID_WYCIECZKI = id;

    IF nowe_miejsca < 0 OR test > nowe_miejsca THEN
        raise_application_error(-20014, 'Podana liczba miejsc jest za mała');
    END IF;

    UPDATE WYCIECZKI
    SET LICZBA_MIEJSC = nowe_miejsca
    WHERE ID_WYCIECZKI = id;
END;

```

Zastosowanie triggerow pozwoliło na tworzenie czytelniejszych oraz krótszych procedur.