

# Ćwiczenie 4.

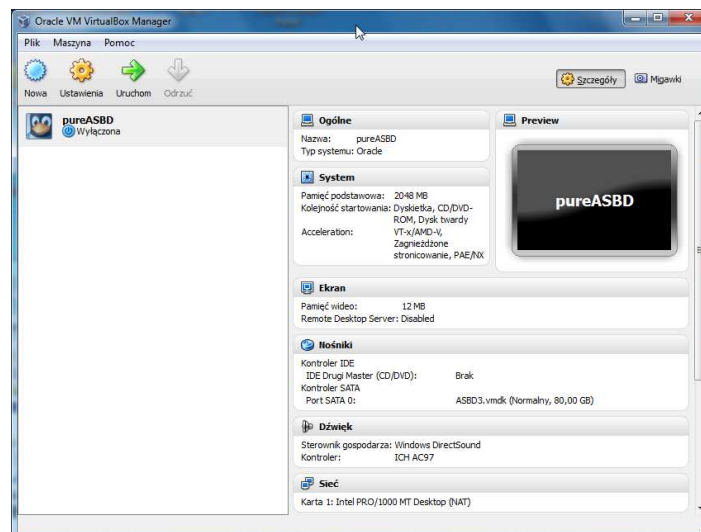
## Opcja obiektowa bazy danych

### 1. Uruchomienie i skonfigurowanie środowiska do ćwiczeń

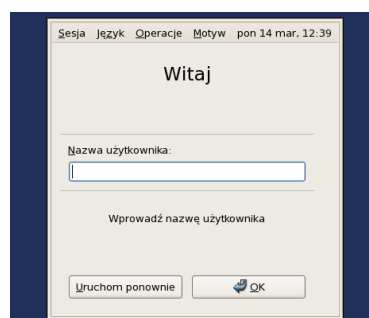
Czas trwania: 15 minut

Zadaniem niniejszych ćwiczeń jest przedstawienie podstawowych zagadnień dotyczących wykorzystywania funkcji analitycznych w języku SQL.

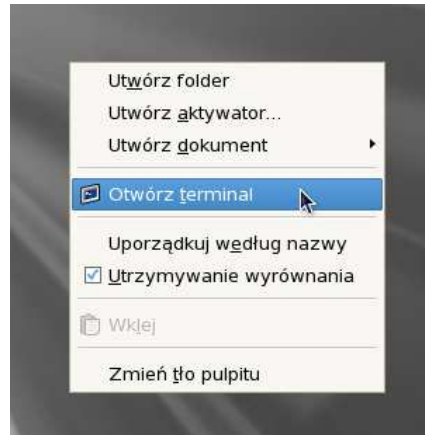
1. Uruchom środowisko wirtualizacji – kliknij na umieszczoną na pulpicie komputera-gospodarza ikonę *Oracle VM VirtualBox*.



2. Spróbuj uruchomić maszynę wirtualną. W tym celu zaznacz w lewym panelu środowiska pozycję **ASBD** i naciśnij umieszczony na pasku narzędzi przycisk **Uruchom** (możesz również wybrać pozycję **Uruchom** z menu kontekstowego, dostępnego po kliknięciu prawym klawiszem myszy na pozycji **ASBD**).
3. Po pomyślnym uruchomieniu maszyny wirtualnej powinien zostać wyświetlony ekran logowania (jeśli ekran maszyny wirtualnej nie zajmuje całego ekranu komputera-gospodarza, użyj odpowiedniego skrótu klawiszowego aby to zmienić).



4. Zaloguj się do systemu operacyjnego maszyny wirtualnej jako użytkownik *Oracle* z hasłem *oracle*. Podaj powyższe informacje i naciśnij przycisk **OK**.
5. Uruchom terminal. Można to wykonać przez kliknięcie prawym klawiszem myszki na pulpit i wybranie z menu kontekstowego pozycji **Otwórz terminal**.



6. Sprawdź wartości zmiennych środowiskowych dotyczących instancji *Oracle*. Wykorzystaj w tym celu polecenie: `set | grep ORACLE`

```
oracle@localhost:~  
Plik Edycja Widok Terminal Zakładki Pomoc  
[oracle@localhost ~]$ set | grep ORACLE  
ORACLE_BASE=/u01/app/oracle  
ORACLE_HOME=/u01/app/oracle/product/11.2.0/dbhome_1  
ORACLE_SID=baza01  
[oracle@localhost ~]$
```

7. Jeśli zmienne środowiskowe nie są ustawione poprawnie, to wczytaj ich wartości z pliku `.bashrc` umieszczonym w Twoim katalogu domowym

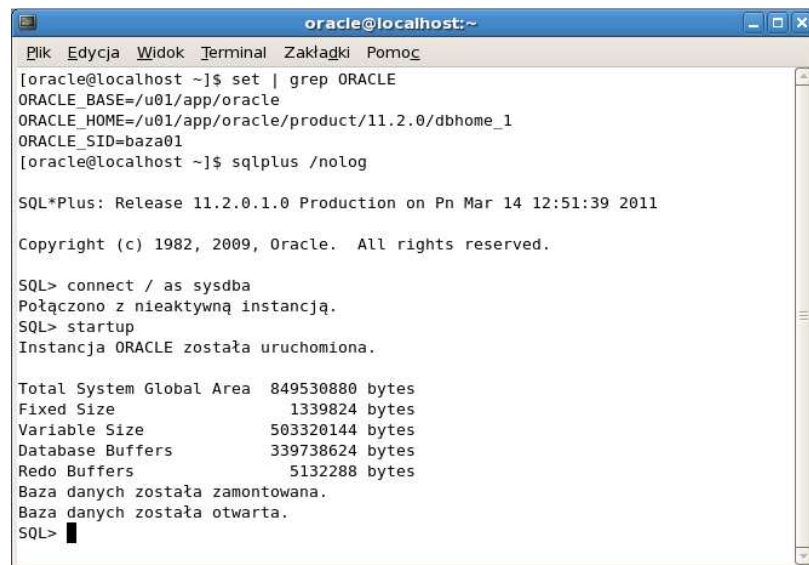
```
[oracle@localhost ~] source .bashrc
```

8. Jeśli zmienne środowiskowe są ustawione poprawnie, uruchom program *sqlplus*. Wykorzystaj opcję *nolog*: `sqlplus /nolog`

```
oracle@localhost:~  
Plik Edycja Widok Terminal Zakładki Pomoc  
[oracle@localhost ~]$ set | grep ORACLE  
ORACLE_BASE=/u01/app/oracle  
ORACLE_HOME=/u01/app/oracle/product/11.2.0/dbhome_1  
ORACLE_SID=baza01  
[oracle@localhost ~]$ sqlplus /nolog  
  
SQL*Plus: Release 11.2.0.1.0 Production on Pn Mar 14 12:51:39 2011  
Copyright (c) 1982, 2009, Oracle. All rights reserved.  
  
SQL>
```

9. Następnie zaloguj się korzystając z autoryzacji użytkownika administracyjnego przez system operacyjny. Wykonaj w tym celu polecenie: `connect / as sysdba`

Następnie uruchom bazę danych poleceniem **startup**



```
oracle@localhost:~  
Plik Edycja Widok Terminal Zakładki Pomoc  
[oracle@localhost ~]$ set | grep ORACLE  
ORACLE_BASE=/u01/app/oracle  
ORACLE_HOME=/u01/app/oracle/product/11.2.0/dbhome_1  
ORACLE_SID=baza01  
[oracle@localhost ~]$ sqlplus /nolog  
  
SQL*Plus: Release 11.2.0.1.0 Production on Pn Mar 14 12:51:39 2011  
  
Copyright (c) 1982, 2009, Oracle. All rights reserved.  
  
SQL> connect / as sysdba  
Połączono z nieaktywną instancją.  
SQL> startup  
Instancja ORACLE została uruchomiona.  
  
Total System Global Area 849530880 bytes  
Fixed Size 1339824 bytes  
Variable Size 503320144 bytes  
Database Buffers 339738624 bytes  
Redo Buffers 5132288 bytes  
Baza danych została zamontowana.  
Baza danych została otwarta.  
SQL>
```

10. Zaloguj się jako użytkownik scott z hasłem tiger.

## 2. Tworzenie obiektów

Czas trwania: 60 minut

Przystąpimy teraz do przeglądu mechanizmów obiektowych oferowanych przez RDBMS Oracle

1. Utwórz typ obiektowy PRACOWNIK.

```
CREATE TYPE Pracownik AS OBJECT (  
    nazwisko VARCHAR2(20),  
    pensja    NUMBER(6,2),  
    etat      VARCHAR2(15),  
    data_ur   DATE  
);  
/
```

2. Utwórz tabelę obiektową i wstaw do niej obiekt reprezentujący pracownika Kowalskiego

```
CREATE TABLE PracownicyObjTab OF PRACOWNIK;  
  
INSERT INTO PracownicyObjTab VALUES (  
    NEW Pracownik('Kowalski',2500,'ASYSTENT',DATE '1965-07-01'));
```

3. Wyświetl zawartość tabeli obiektowej w trybie dostępu relacyjnego i obiektowego.

```
SELECT * FROM PracownicyObjTab;  
  
SELECT VALUE(p) FROM PracownicyObjTab p;
```

4. Utwórz tabelę, która będzie zawierała obiektowy atrybut. Wstaw do tabeli krotkę zawierającą obiekt.

```
CREATE TABLE ProjektyTab (  
    symbol CHAR(6),  
    nazwa VARCHAR(100),  
    budzet NUMBER,  
    kierownik Pracownik);  
  
INSERT INTO ProjektyTab VALUES  
('AB 001','Projekt X',20000,  
    NEW Pracownik('Nowak',3200,'ADIUNKT',null));
```

5. Wyświetl zawartość nowoutworzonej tabeli. Sprawdź, jak funkcjonuje dostęp do składowych obiektów za pomocą notacji kropkowej.

```
SELECT nazwa, kierownik FROM ProjektyTab;  
  
SELECT p.kierownik.nazwisko FROM ProjektyTab p;
```

6. Dodaj do definicji typu Pracownik funkcję wyznaczającą wiek pracownika oraz metodę służącą do przyznania pracownikowi podwyżki

```
ALTER TYPE Pracownik REPLACE AS OBJECT (  
    nazwisko VARCHAR2(20),  
    pensja    NUMBER(6,2),  
    etat      VARCHAR2(15),  
    data_ur   DATE,  
    MEMBER FUNCTION wiek RETURN NUMBER,  
    MEMBER PROCEDURE podwyzka(p_kwota NUMBER) );  
/  
  
CREATE OR REPLACE TYPE BODY Pracownik AS  
    MEMBER FUNCTION wiek RETURN NUMBER IS  
    BEGIN  
        RETURN EXTRACT (YEAR FROM CURRENT_DATE) - EXTRACT (YEAR FROM  
data_ur);  
    END wiek;  
  
    MEMBER PROCEDURE podwyzka(p_kwota NUMBER) IS  
    BEGIN  
        pensja := pensja + p_kwota;  
    END podwyzka;  
END;  
/
```

7. Wyświetl wiek pracowników umieszczonych w tabeli PracownicyObjTab. Następnie, przyznaj 200 zł podwyżki kierownikowi projektu 'AB 001' (tabela ProjektyTab)

```
SELECT p.nazwisko, p.data_ur, p.wiek() FROM PracownicyObjTab p;  
  
DECLARE  
    l_kierownik Pracownik;  
BEGIN  
    SELECT kierownik INTO l_kierownik  
    FROM ProjektyTab WHERE symbol = 'AB 001';  
  
    l_kierownik.podwyzka(200);  
  
    UPDATE ProjektyTab  
    SET kierownik = l_kierownik  
    WHERE symbol = 'AB 001';  
END;
```

8. Wyświetl unikalne identyfikatory (OIDs) obiektów przechowywanych w tabeli PracownicyObjTab.

```
SELECT VALUE(p), REF(p) FROM PracownicyObjTab p;
```

9. Dodaj do typu obiektowego Pracownik metodę, za pomocą której będzie można porównywać ze sobą pracowników korzystając z ich pensji i wieku.

```
ALTER TYPE Pracownik ADD MAP MEMBER FUNCTION odwzoruj  
RETURN NUMBER CASCADE INCLUDING TABLE DATA;
```

```
CREATE OR REPLACE TYPE BODY Pracownik AS  
  MEMBER FUNCTION wiek RETURN NUMBER IS  
  BEGIN  
    RETURN EXTRACT (YEAR FROM CURRENT_DATE) -  
      EXTRACT (YEAR FROM data_ur);  
  END wiek;  
  MEMBER PROCEDURE podwyzka(p_kwota NUMBER) IS  
  BEGIN  
    pensja := pensja + p_kwota;  
  END podwyzka;  
  MAP MEMBER FUNCTION odwzoruj RETURN NUMBER IS  
  BEGIN  
    RETURN ROUND(pensja,-3) + wiek();  
  END odwzoruj;  
END;  
/
```

10. Dodaj do tabeli PracownicyObjTab dwóch nowych pracowników i sprawdź działanie funkcji odwzorowującej.

```
INSERT INTO PracownicyObjTab VALUES  
  (NEW Pracownik('Nowak',2000,'ADIUNKT',DATE '1961-02-15'));  
INSERT INTO PracownicyObjTab VALUES  
  (NEW Pracownik('Janiak',1800,'ASYSTENT',DATE '1973-12-02'));  
  
SELECT p.nazwisko, p.pensja, p.wiek()  
FROM PracownicyObjTab p  
ORDER BY VALUE(p);  
  
SELECT * FROM PracownicyObjTab p  
WHERE VALUE(p) > ( SELECT VALUE(r) FROM PracownicyObjTab r  
                   WHERE r.nazwisko = 'Janiak' );
```

11. Do definicji typu obiektowego Pracownik dodaj konstruktor. W ciele typu obiektowego umieść następnie implementację konstruktora.

```
ALTER TYPE Pracownik REPLACE AS OBJECT (  
    nazwisko VARCHAR2(20),  
    pensja    NUMBER(6,2),  
    etat      VARCHAR2(15),  
    data_ur   DATE,  
    MEMBER FUNCTION wiek RETURN NUMBER,  
    MEMBER PROCEDURE podwyzka(p_kwota NUMBER),  
    MAP MEMBER FUNCTION odwzoruj RETURN NUMBER,  
    CONSTRUCTOR FUNCTION Pracownik(p_nazwisko VARCHAR2)  
        RETURN SELF AS RESULT );  
  
CREATE OR REPLACE TYPE BODY Pracownik AS  
    MEMBER FUNCTION wiek RETURN NUMBER IS  
    BEGIN  
        RETURN EXTRACT (YEAR FROM CURRENT_DATE) -  
            EXTRACT (YEAR FROM data_ur);  
    END wiek;  
    MEMBER PROCEDURE podwyzka(p_kwota NUMBER) IS  
    BEGIN  
        pensja := pensja + p_kwota;  
    END podwyzka;  
    MAP MEMBER FUNCTION odwzoruj RETURN NUMBER IS  
    BEGIN  
        RETURN ROUND(pensja,-3) + wiek();  
    END odwzoruj;  
  
    CONSTRUCTOR FUNCTION Pracownik(p_nazwisko VARCHAR2)  
        RETURN SELF AS RESULT IS  
    BEGIN  
        SELF.nazwisko := p_nazwisko; SELF.pensja := 1000;  
        SELF.etat := null; SELF.data_ur := null;  
        RETURN;  
    END;  
END;  
/
```

12. Przetestuj działanie konstruktora tworząc nowy obiekt i wstawiając go do tabeli PracownicyObjTab.

```
INSERT INTO PracownicyObjTab VALUES (  
    NEW Pracownik('Dziamdziak') );
```

13. Utwórz typy obiektowe Osoba i Adres. Wykorzystaj referencje do powiązania osób z adresami zamieszkania. Następnie, utwórz tabele obiektowe i wypełnij je przykładowymi danymi.

```
CREATE TYPE Adres AS OBJECT (  
    ulica      VARCHAR2(15),  
    dom        NUMBER(4),  
    mieszkanie NUMBER(3));  
/  
  
CREATE TYPE Osoba AS OBJECT (  
    Nazwisko VARCHAR2(20),  
    Imie      VARCHAR2(15),  
    GdzieMieszka REF Adres);  
/  
  
CREATE TABLE AdresyObjTab OF Adres;  
CREATE TABLE OsobyObjTab OF Osoba;  
  
ALTER TABLE OsobyObjTab ADD  
SCOPE FOR(GdzieMieszka) IS AdresyObjTab;  
  
INSERT INTO AdresyObjTab VALUES  
    (NEW Adres('Kolejowa',2,18));  
INSERT INTO OsobyObjTab VALUES  
    (NEW Osoba('Kowalska','Anna',null));  
INSERT INTO OsobyObjTab VALUES  
    (NEW Osoba('Kowalski','Jan',null));  
  
UPDATE OsobyObjTab o  
SET o.gdziemieszka = (  
    SELECT REF(a) FROM AdresyObjTab a  
    WHERE a.ulica = 'Kolejowa' );
```

14. Sprawdź różne sposoby wykorzystania nawigacji po referencjach.

```
SELECT o.imie, o.nazwisko, Deref(o.gdziemieszka)  
FROM OsobyObjTab o;  
  
SELECT o.imie, o.nazwisko, o.gdziemieszka.ulica,  
       o.gdziemieszka.dom  
FROM OsobyObjTab o;  
  
SELECT o.imie, o.nazwisko, a.ulica, a.dom, a.mieszkanie  
FROM OsobyObjTab o JOIN AdresyObjTab a  
    ON o.gdziemieszka = REF(a);
```

15. Zaobserwuj zjawisko wiszących referencji.

```
DELETE FROM AdresyObjTab a WHERE a.ulica = 'Kolejowa';  
  
SELECT * FROM OsobyObjTab o WHERE o.gdziemieszka IS NULL;  
  
SELECT * FROM OsobyObjTab o WHERE o.gdziemieszka IS Dangling;
```



### 3. Opcja obiektowa: ćwiczenia samodzielne

#### Tworzenie typów obiektowych

1. Zdefiniuj typ obiektowy reprezentujący SAMOCHODY. Każdy samochód powinien mieć markę, model, liczbę kilometrów oraz datę produkcji i cenę. Stwórz tablicę obiektową i wprowadź kilka przykładowych obiektów, obejrzyj zawartość tablicy

```
SQL> desc samochod
```

Nazwa	Wartość NULL?	Typ
MARKA		VARCHAR2(20)
MODEL		VARCHAR2(20)
KILOMETRY		NUMBER
DATA_PRODUKCJI		DATE
CENA		NUMBER(10,2)

```
SQL> select * from samochody;
```

MARKA	MODEL	KILOMETRY	DATA_PRODU	CENA
FIAT	BRAVA	60000	30-11-1999	25000
FORD	MONDEO	80000	10-05-1997	45000
MAZDA	323	12000	22-09-2000	52000

2. Stwórz tablicę WLASCICIELE zawierającą imiona i nazwiska właścicieli oraz atrybut obiektowy SAMOCHOD. Wprowadź do tabeli przykładowe dane i wyświetl jej zawartość.

```
SQL> desc wlasciciele
```

Nazwa	Wartość NULL?	Typ
IMIE		VARCHAR2(100)
NAZWISKO		VARCHAR2(100)
AUTO		SAMOCHOD

```
SQL> select * from wlasciciele;
```

IMIE	NAZWISKO	AUTO(MARKA, MODEL, KILOMETRY, DATA_PRODUKCJI, CENA)
JAN	KOWALSKI	SAMOCHOD('FIAT', 'SEICENTO', 30000, '02-12-0010', 19500)
ADAM	NOWAK	SAMOCHOD('OPEL', 'ASTRA', 34000, '01-06-0009', 33700)

3. Wartość samochodu maleje o 10% z każdym rokiem. Dodaj do typu obiektowego SAMOCHOD metodę wyliczającą na podstawie wieku i przebiegu aktualną wartość samochodu.

```
SQL> SELECT s.marka, s.cena, s.wartosc() FROM SAMOCHODY s;
```

MARKA	CENA	S.WARTOSC()
FIAT	25000	12500
FORD	45000	9000
MAZDA	52000	26000

4. Dodaj do typu SAMOCHOD metodę odwzorowującą, która pozwoli na porównywanie samochodów na podstawie ich wieku i zużycia. Przyjmij, że 10000 km odpowiada jednemu rokowi wieku samochodu.

```
SQL> SELECT * FROM SAMOCHODY s ORDER BY VALUE(s);
```

MARKA	MODEL	KILOMETRY	DATA_PRODU	CENA
MAZDA	323	12000	22-09-2000	52000
FIAT	BRAVA	60000	30-11-1999	25000
FORD	MONDEO	80000	10-05-1997	45000

5. Stwórz typ WLASCICIEL zawierający imię i nazwisko właściciela samochodu, dodaj do typu SAMOCHOD referencje do właściciela. Wypełnij tabelę przykładowymi danymi.

## Kolekcje

6. Zbuduj kolekcję (tablicę o zmiennym rozmiarze) zawierającą informacje o przedmiotach (łańcuchy znaków). Wstaw do kolekcji przykładowe przedmioty, rozszerz kolekcję, wyświetl zawartość kolekcji, usuń elementy z końca kolekcji

```
DECLARE
```

```
    TYPE t_przedmioty IS VARRAY(10) OF VARCHAR2(20);  
    moje_przedmioty t_przedmioty := t_przedmioty('');
```

```
BEGIN
```

```
    moje_przedmioty(1) := 'MATEMATYKA';  
    moje_przedmioty.EXTEND(9);
```

```
    FOR i IN 2..10 LOOP  
        moje_przedmioty(i) := 'PRZEDMIOT_' || i;  
    END LOOP;
```

```
    FOR i IN moje_przedmioty.FIRST()..moje_przedmioty.LAST() LOOP  
        DBMS_OUTPUT.PUT_LINE(moje_przedmioty(i));  
    END LOOP;
```

```
    moje_przedmioty.TRIM(2);
```

```
    FOR i IN moje_przedmioty.FIRST()..moje_przedmioty.LAST() LOOP  
        DBMS_OUTPUT.PUT_LINE(moje_przedmioty(i));  
    END LOOP;
```

```
    DBMS_OUTPUT.PUT_LINE('Limit: ' || moje_przedmioty.LIMIT());  
    DBMS_OUTPUT.PUT_LINE('Liczba elementow: ' || moje_przedmioty.COUNT());
```

```
    moje_przedmioty.EXTEND();  
    moje_przedmioty(9) := 9;
```

```
    DBMS_OUTPUT.PUT_LINE('Limit: ' || moje_przedmioty.LIMIT());  
    DBMS_OUTPUT.PUT_LINE('Liczba elementow: ' || moje_przedmioty.COUNT());
```

```
    moje_przedmioty.DELETE();
```

```
    DBMS_OUTPUT.PUT_LINE('Limit: ' || moje_przedmioty.LIMIT());  
    DBMS_OUTPUT.PUT_LINE('Liczba elementow: ' || moje_przedmioty.COUNT());  
END;
```

7. Zdefiniuj kolekcję (w oparciu o tablicę o zmiennym rozmiarze) zawierającą listę tytułów książek. Wykonaj na kolekcji kilka czynności (rozszerz, usuń jakiś element, wstaw nową książkę)
8. Zbuduj kolekcję (tablicę zagnieżdżoną) zawierającą informacje o wykładowcach. Przetestuj działanie kolekcji podobnie jak w przykładzie 6.

```
DECLARE
  TYPE t_wykladowcy IS TABLE OF VARCHAR2(20);
  moi_wykladowcy t_wykladowcy := t_wykladowcy();

BEGIN
  moi_wykladowcy.EXTEND(2);

  moi_wykladowcy(1) := 'MORZY';
  moi_wykladowcy(2) := 'WOJCIECHOWSKI';

  moi_wykladowcy.EXTEND(8);

  FOR i IN 3..10 LOOP
    moi_wykladowcy(i) := 'WYKLADOWCA_' || i;
  END LOOP;

  FOR i IN moi_wykladowcy.FIRST()..moi_wykladowcy.LAST() LOOP
    DBMS_OUTPUT.PUT_LINE(moi_wykladowcy(i));
  END LOOP;

  moi_wykladowcy.TRIM(2);

  FOR i IN moi_wykladowcy.FIRST()..moi_wykladowcy.LAST() LOOP
    DBMS_OUTPUT.PUT_LINE(moi_wykladowcy(i));
  END LOOP;

  moi_wykladowcy.DELETE(5,7);

  DBMS_OUTPUT.PUT_LINE('Limit: ' || moi_wykladowcy.LIMIT());
  DBMS_OUTPUT.PUT_LINE('Liczba elementow: ' || moi_wykladowcy.COUNT());

  FOR i IN moi_wykladowcy.FIRST()..moi_wykladowcy.LAST() LOOP
    IF moi_wykladowcy.EXISTS(i) THEN
      DBMS_OUTPUT.PUT_LINE(moi_wykladowcy(i));
    END IF;
  END LOOP;

  moi_wykladowcy(5) := 'ZAKRZEWICZ';
  moi_wykladowcy(6) := 'KROLIKOWSKI';
  moi_wykladowcy(7) := 'KOSZLAJDA';

  FOR i IN moi_wykladowcy.FIRST()..moi_wykladowcy.LAST() LOOP
    IF moi_wykladowcy.EXISTS(i) THEN
      DBMS_OUTPUT.PUT_LINE(moi_wykladowcy(i));
    END IF;
  END LOOP;

  DBMS_OUTPUT.PUT_LINE('Limit: ' || moi_wykladowcy.LIMIT());
  DBMS_OUTPUT.PUT_LINE('Liczba elementow: ' || moi_wykladowcy.COUNT());
END;
```

9. Zbuduj kolekcję (w oparciu o tablicę zagnieżdżoną) zawierającą listę miesięcy. Wstaw do kolekcji właściwe dane, usuń parę miesięcy, wyświetl zawartość kolekcji.

## 10. Sprawdź działanie obu rodzajów kolekcji w przypadku atrybutów bazodanowych.

```
CREATE TYPE jezyki_obce AS VARRAY(10) OF VARCHAR2(20);
/

CREATE TYPE stypendium AS OBJECT (
    nazwa VARCHAR2(50),
    kraj VARCHAR2(30),
    jezyki jezyki_obce );
/

CREATE TABLE stypendia OF stypendium;

INSERT INTO stypendia VALUES
('SOKRATES','FRANCJA',jezyki_obce('ANGIELSKI','FRANCUSKI','NIEMIECKI'));
INSERT INTO stypendia VALUES
('ERASMUS','NIEMCY',jezyki_obce('ANGIELSKI','NIEMIECKI','HISZPANSKI'));

SELECT * FROM stypendia;

SELECT s.jezyki FROM stypendia s;

UPDATE STYPENDIA
SET jezyki = jezyki_obce('ANGIELSKI','NIEMIECKI','HISZPANSKI','FRANCUSKI')
WHERE nazwa = 'ERASMUS';

CREATE TYPE lista_egzaminow AS TABLE OF VARCHAR2(20);
/

CREATE TYPE semestr AS OBJECT (
    numer NUMBER,
    egzaminy lista_egzaminow );
/

CREATE TABLE semestry OF semestr
NESTED TABLE egzaminy STORE AS tab_egzaminy;

INSERT INTO semestry VALUES
(semestr(1,lista_egzaminow('MATEMATYKA','LOGIKA','ALGEBRA')));
INSERT INTO semestry VALUES
(semestr(2,lista_egzaminow('BAZY DANYCH','SYSTEMY OPERACYJNE')));

SELECT s.numer, e.*
FROM semestry s, TABLE(s.egzaminy) e;

SELECT e.*
FROM semestry s, TABLE ( s.egzaminy ) e;

SELECT * FROM TABLE ( SELECT s.egzaminy FROM semestry s WHERE numer=1 );

INSERT INTO TABLE ( SELECT s.egzaminy FROM semestry s WHERE numer=2 )
VALUES ( 'METODY NUMERYCZNE' );

UPDATE TABLE ( SELECT s.egzaminy FROM semestry s WHERE numer=2 ) e
SET e.column_value = 'SYSTEMY ROZPROSZONE'
WHERE e.column_value = 'SYSTEMY OPERACYJNE';

DELETE FROM TABLE ( SELECT s.egzaminy FROM semestry s WHERE numer=2 ) e
WHERE e.column_value = 'BAZY DANYCH';
```

## 11. Zbuduj tabelę ZAKUPY zawierającą atrybut zbiorowy KOSZYK\_PRODUKTOW w postaci tabeli zagnieżdżonej. Wstaw do tabeli przykładowe dane. Wyświetl zawartość tabeli, usuń wszystkie transakcje zawierające wybrany produkt.

## Polimorfizm, dziedziczenie, perspektywy obiektowe

### 12. Zbuduj hierarchię reprezentującą instrumenty muzyczne

```
CREATE TYPE instrument AS OBJECT (  
    nazwa VARCHAR2(20),  
    dzwiek VARCHAR2(20),  
    MEMBER FUNCTION graj RETURN VARCHAR2 ) NOT FINAL;  
  
CREATE TYPE BODY instrument AS  
    MEMBER FUNCTION graj RETURN VARCHAR2 IS  
    BEGIN  
        RETURN dzwiek;  
    END;  
END;  
/  
  
CREATE TYPE instrument_dety UNDER instrument (  
    material VARCHAR2(20),  
    OVERRIDING MEMBER FUNCTION graj RETURN VARCHAR2,  
    MEMBER FUNCTION graj(glosnosc VARCHAR2) RETURN VARCHAR2 );  
  
CREATE OR REPLACE TYPE BODY instrument_dety AS  
    OVERRIDING MEMBER FUNCTION graj RETURN VARCHAR2 IS  
    BEGIN  
        RETURN 'dmucham: ' || dzwiek;  
    END;  
    MEMBER FUNCTION graj(glosnosc VARCHAR2) RETURN VARCHAR2 IS  
    BEGIN  
        RETURN glosnosc || ':' || dzwiek;  
    END;  
END;  
/  
  
CREATE TYPE instrument_klawiszowy UNDER instrument (  
    producent VARCHAR2(20),  
    OVERRIDING MEMBER FUNCTION graj RETURN VARCHAR2 );  
  
CREATE OR REPLACE TYPE BODY instrument_klawiszowy AS  
    OVERRIDING MEMBER FUNCTION graj RETURN VARCHAR2 IS  
    BEGIN  
        RETURN 'stukam w klawisze: ' || dzwiek;  
    END;  
END;  
/  
  
DECLARE  
    tamburyn instrument := instrument('tamburyn','brzdek-brzdek');  
    trabka instrument_dety := instrument_dety('trabka','tra-ta-ta','metalowa');  
    fortepian instrument_klawiszowy := instrument_klawiszowy('fortepian','ping-  
ping','steinway');  
BEGIN  
    dbms_output.put_line(tamburyn.graj);  
    dbms_output.put_line(trabka.graj);  
    dbms_output.put_line(trabka.graj('glosno'));  
    dbms_output.put_line(forteplan.graj);  
END;
```

### 13. Zbuduj hierarchię zwierząt i przetestuj klasy abstrakcyjne

```
CREATE TYPE istota AS OBJECT (  
    nazwa VARCHAR2(20),  
    NOT INSTANTIABLE MEMBER FUNCTION poluj(ofiara CHAR) RETURN CHAR )  
    NOT INSTANTIABLE NOT FINAL;  
  
CREATE TYPE lew UNDER istota (  
    liczba_nog NUMBER,  
    OVERRIDING MEMBER FUNCTION poluj(ofiara CHAR) RETURN CHAR );  
  
CREATE OR REPLACE TYPE BODY lew AS  
    OVERRIDING MEMBER FUNCTION poluj(ofiara CHAR) RETURN CHAR IS  
    BEGIN  
        RETURN 'upolowana ofiara: '||ofiara;  
    END;  
END;  
  
DECLARE  
    KrolLew lew := lew('LEW',4);  
    InnaIstota istota := istota('JAKIES ZWIERZE');  
BEGIN  
    DBMS_OUTPUT.PUT_LINE( KrolLew.poluj('antylopa') );  
END;
```

### 14. Zbadaj własność polimorfizmu na przykładzie hierarchii instrumentów

```
DECLARE  
    tamburyn instrument;  
    cymbalki instrument;  
    trabka instrument_dety;  
    saksofon instrument_dety;  
BEGIN  
    tamburyn := instrument('tamburyn','brzdek-brzdek');  
    cymbalki := instrument_dety('cymbalki','ding-ding','metalowe');  
    trabka := instrument_dety('trabka','tra-ta-ta','metalowa');  
    -- saksofon := instrument('saksofon','tra-taaaa');  
    -- saksofon := TREAT( instrument('saksofon','tra-taaaa') AS instrument_dety);  
END;
```

### 15. Zbuduj tabelę zawierającą różne instrumenty. Zbadaj działanie funkcji wirtualnych

```
CREATE TABLE instrumenty OF instrument;  
  
INSERT INTO instrumenty VALUES ( instrument('tamburyn','brzdek-brzdek') );  
INSERT INTO instrumenty VALUES ( instrument_dety('trabka','tra-ta-ta','metalowa') );  
INSERT INTO instrumenty VALUES ( instrument_klawiszowy('fortepian','ping-ping','steinway') );  
  
SELECT i.nazwa, i.graj() FROM instrumenty i;
```

**16. Utwórz dodatkową tabelę PRZEDMIOTY i wypełnij ją przykładowymi danymi**

```
CREATE TABLE PRZEDMIOTY (  
    NAZWA VARCHAR2(50),  
    NAUCZYCIEL NUMBER REFERENCES PRACOWNICY(ID_PRAC)  
);  
  
INSERT INTO PRZEDMIOTY VALUES ('BAZY DANYCH',100);  
INSERT INTO PRZEDMIOTY VALUES ('SYSTEMY OPERACYJNE',100);  
INSERT INTO PRZEDMIOTY VALUES ('PROGRAMOWANIE',110);  
INSERT INTO PRZEDMIOTY VALUES ('SIECI KOMPUTEROWE',110);  
INSERT INTO PRZEDMIOTY VALUES ('BADANIA OPERACYJNE',120);  
INSERT INTO PRZEDMIOTY VALUES ('GRAFIKA KOMPUTEROWA',120);  
INSERT INTO PRZEDMIOTY VALUES ('BAZY DANYCH',130);  
INSERT INTO PRZEDMIOTY VALUES ('SYSTEMY OPERACYJNE',140);  
INSERT INTO PRZEDMIOTY VALUES ('PROGRAMOWANIE',140);  
INSERT INTO PRZEDMIOTY VALUES ('SIECI KOMPUTEROWE',140);  
INSERT INTO PRZEDMIOTY VALUES ('BADANIA OPERACYJNE',150);  
INSERT INTO PRZEDMIOTY VALUES ('GRAFIKA KOMPUTEROWA',150);  
INSERT INTO PRZEDMIOTY VALUES ('BAZY DANYCH',160);  
INSERT INTO PRZEDMIOTY VALUES ('SYSTEMY OPERACYJNE',160);  
INSERT INTO PRZEDMIOTY VALUES ('PROGRAMOWANIE',170);  
INSERT INTO PRZEDMIOTY VALUES ('SIECI KOMPUTEROWE',180);  
INSERT INTO PRZEDMIOTY VALUES ('BADANIA OPERACYJNE',180);  
INSERT INTO PRZEDMIOTY VALUES ('GRAFIKA KOMPUTEROWA',190);  
INSERT INTO PRZEDMIOTY VALUES ('GRAFIKA KOMPUTEROWA',200);  
INSERT INTO PRZEDMIOTY VALUES ('GRAFIKA KOMPUTEROWA',210);  
INSERT INTO PRZEDMIOTY VALUES ('PROGRAMOWANIE',220);  
INSERT INTO PRZEDMIOTY VALUES ('SIECI KOMPUTEROWE',220);  
INSERT INTO PRZEDMIOTY VALUES ('BADANIA OPERACYJNE',230);
```

**17. Stwórz typ który będzie odpowiadał krotkom z relacji ZESPOLY**

```
CREATE TYPE ZESPOL AS OBJECT (  
    ID_ZESP NUMBER,  
    NAZWA VARCHAR2(50),  
    ADRES VARCHAR2(100)  
);  
/
```

**18. Na bazie stworzonego typu zbuduj perspektywę obiektową przedstawiającą dane z relacji ZESPOLY w sposób obiektowy.**

```
CREATE OR REPLACE VIEW ZESPOLY_V OF ZESPOL  
WITH OBJECT IDENTIFIER(ID_ZESP)  
AS SELECT ID_ZESP, NAZWA, ADRES FROM ZESPOLY;
```

19. Utwórz typ tablicowy do przechowywania zbioru przedmiotów wykładanych przez każdego nauczyciela. Stwórz typ odpowiadający krotkom z relacji PRACOWNICY. Każdy obiekt typu pracownik powinien posiadać unikalny numer, nazwisko, etat, datę zatrudnienia, płacę podstawową, miejsce pracy (referencja do właściwego zespołu) oraz zbiór wykładanych przedmiotów. Typ powinien też zawierać metodę służącą do wyliczania liczby przedmiotów wykładanych przez wykładowcę.

```
CREATE TYPE PRZEDMIOTY_TAB AS TABLE OF VARCHAR2(100);  
/
```

```
CREATE TYPE PRACOWNIK AS OBJECT (  
    ID_PRAC NUMBER,  
    NAZWISKO VARCHAR2(30),  
    ETAT VARCHAR2(20),  
    ZATRUDNIONY DATE,  
    PLACA_POD NUMBER(10,2),  
    MIEJSCE_PRACY REF ZESPOL,  
    PRZEDMIOTY PRZEDMIOTY_TAB,  
    MEMBER FUNCTION ILE_PRZEDMIOTOW RETURN NUMBER  
);  
/
```

```
CREATE OR REPLACE TYPE BODY PRACOWNIK AS  
    MEMBER FUNCTION ILE_PRZEDMIOTOW RETURN NUMBER IS  
    BEGIN  
        RETURN PRZEDMIOTY.COUNT();  
    END ILE_PRZEDMIOTOW;  
END;
```

20. Na bazie stworzonego typu zbuduj perspektywę obiektową przedstawiającą dane z relacji PRACOWNICY w sposób obiektowy.

```
CREATE OR REPLACE VIEW PRACOWNICY_V OF PRACOWNIK  
WITH OBJECT IDENTIFIER (ID_PRAC)  
AS SELECT ID_PRAC, NAZWISKO, ETAT, ZATRUDNIONY, PLACA_POD,  
    MAKE_REF(ZESPOLY_V, ID_ZESP),  
    CAST(MULTISET( SELECT NAZWA FROM PRZEDMIOTY WHERE NAUCZYCIEL=P.ID_PRAC ) AS  
PRZEDMIOTY_TAB )  
FROM PRACOWNICY P;
```

21. Sprawdź różne sposoby wyświetlania danych z perspektywy obiektowej

```
SELECT *  
FROM PRACOWNICY_V;
```

```
SELECT P.NAZWISKO, P.ETAT, P.MIEJSCE_PRACY.NAZWA  
FROM PRACOWNICY_V P;
```

```
SELECT P.NAZWISKO, P.ILE_PRZEDMIOTOW()  
FROM PRACOWNICY_V P;
```

```
SELECT *  
FROM TABLE( SELECT PRZEDMIOTY FROM PRACOWNICY_V WHERE NAZWISKO='WEGLARZ' );
```

```
SELECT NAZWISKO, CURSOR( SELECT PRZEDMIOTY  
FROM PRACOWNICY_V  
WHERE ID_PRAC=P.ID_PRAC)  
FROM PRACOWNICY_V P;
```



22. Dane są poniższe relacje. Zbuduj interfejs składający się z dwóch typów i dwóch perspektyw obiektowych, który umożliwi interakcję ze schematem relacyjnym. Typ odpowiadający krotkom z relacji PISARZE powinien posiadać metodę wyznaczającą liczbę książek napisanych przez danego pisarza. Typ odpowiadający krotkom z relacji KSIAZKI powinien posiadać metodę wyznaczającą wiek książki (w latach).

```
CREATE TABLE PISARZE (  
    ID_PISARZA NUMBER PRIMARY KEY,  
    NAZWISKO VARCHAR2(20),  
    DATA_UR DATE );  
  
CREATE TABLE KSIAZKI (  
    ID_KSIAZKI NUMBER PRIMARY KEY,  
    ID_PISARZA NUMBER NOT NULL REFERENCES PISARZE,  
    TYTUL VARCHAR2(50),  
    DATA_WYDANIA DATE );  
  
INSERT INTO PISARZE VALUES(10,'SIENKIEWICZ',DATE '1880-01-01');  
INSERT INTO PISARZE VALUES(20,'PRUS',DATE '1890-04-12');  
INSERT INTO PISARZE VALUES(30,'ZEROMSKI',DATE '1899-09-11');  
  
INSERT INTO KSIAZKI(ID_KSIAZKI,ID_PISARZA,TYTUL,DATA_WYDANIA)  
VALUES(10,10,'OGNIEM I MIECZEM',DATE '1990-01-05');  
INSERT INTO KSIAZKI(ID_KSIAZKI,ID_PISARZA,TYTUL,DATA_WYDANIA)  
VALUES(20,10,'POTOP',DATE '1975-12-09');  
INSERT INTO KSIAZKI(ID_KSIAZKI,ID_PISARZA,TYTUL,DATA_WYDANIA)  
VALUES(30,10,'PAN WOŁODYJOWSKI',DATE '1987-02-15');  
INSERT INTO KSIAZKI(ID_KSIAZKI,ID_PISARZA,TYTUL,DATA_WYDANIA)  
VALUES(40,20,'FARAON',DATE '1948-01-21');  
INSERT INTO KSIAZKI(ID_KSIAZKI,ID_PISARZA,TYTUL,DATA_WYDANIA)  
VALUES(50,20,'LALKA',DATE '1994-08-01');  
INSERT INTO KSIAZKI(ID_KSIAZKI,ID_PISARZA,TYTUL,DATA_WYDANIA)  
VALUES(60,30,'PRZEDWIOSNIE',DATE '1938-02-02');
```

23. Zbuduj hierarchię aut (auto, auto osobowe, auto ciężarowe) i przetestuj następujące mechanizmy obiektowe:

- dziedziczenie: auto osobowe ma dodatkowe atrybuty określające liczbę miejsc (atrybut numeryczny) oraz wyposażenie w klimatyzację (tak/nie). Auto ciężarowe ma dodatkowy atrybut określający maksymalną ładowność (atrybut numeryczny)
- przesłanianie metod: zdefiniuj na nowo w typach AUTO\_OSOBOWE i AUTO\_CIEZAROWE metodę określającą wartość auta. W przypadku auta osobowego przyjmij, że fakt wyposażenia auta w klimatyzację zwiększa wartość auta o 50%. W przypadku auta ciężarowego ładowność powyżej 10T zwiększa wartość auta o 100%
- polimorfizm i późne wiązanie metod: wstaw do tabeli obiektowej przechowującej auta dwa auta osobowe (jedno z klimatyzacją i drugie bez klimatyzacji) oraz dwa auta ciężarowe (o ładownościach 8T i 12T).

Wyświetl markę i wartość każdego auta przechowywanego w tabeli obiektowej.

```
CREATE TYPE AUTO AS OBJECT (  
    MARKA VARCHAR2(20),  
    MODEL VARCHAR2(20),  
    KILOMETRY NUMBER,  
    DATA_PRODUKCJI DATE,  
    CENA NUMBER(10,2),  
    MEMBER FUNCTION WARTOSC RETURN NUMBER  
);  
  
CREATE OR REPLACE TYPE BODY AUTO AS  
    MEMBER FUNCTION WARTOSC RETURN NUMBER IS  
        WIEK NUMBER;  
        WARTOSC NUMBER;  
    BEGIN  
        WIEK := ROUND(MONTHS_BETWEEN(SYSDATE, DATA_PRODUKCJI)/12);  
        WARTOSC := CENA - (WIEK * 0.1 * CENA);  
        IF (WARTOSC < 0) THEN  
            WARTOSC := 0;  
        END IF;  
        RETURN WARTOSC;  
    END WARTOSC;  
END;  
  
CREATE TABLE AUTA OF AUTO;  
  
INSERT INTO AUTA VALUES (AUTO('FIAT', 'BRAVA', 60000, DATE '1999-11-30', 25000));  
INSERT INTO AUTA VALUES (AUTO('FORD', 'MONDEO', 80000, DATE '1997-05-10', 45000));  
INSERT INTO AUTA VALUES (AUTO('MAZDA', '323', 12000, DATE '2000-09-22', 52000));
```