

Baze de date dinamice

Ruxandra Stoean
<http://inf.ucv.ro/~rstoean>
ruxandra.stoean@inf.ucv.ro

Baze de date dinamice

- Este evident faptul ca un program Prolog este o bază de date ce conține predicate.
- Până acum, am introdus clauzele pentru predicatele realizate de noi din interiorul programului.
- Prologul ne oferă însă posibilitatea să manevrăm baza de date în mod direct și ne oferă predicate predefinite care fac acest lucru.

Adaugare clauze

- `assert(X)` – adaugă clauza `X` ca ultima clauză a predicatului din care face parte.
- `asserta(X)` – clauza `X` este adăugată ca prima clauză a predicatului.
- `assertz(X)` – același efect cu `assert(X)`.

Retragere clauze

- $\text{retract}(X)$ – scoate clauza X din baza de date.
- $\text{retractall}(X)$ – sunt eliminate toate faptele sau regulile din baza de date pentru care capul clauzei este unificat cu X .

Exemplu

- Presupunem că avem următoarele fapte într-un fișier Prolog:

`copil(ionut).`

`copil(marian).`

`copil(simona).`

- Presupunem că ei sunt introduși în ordinea vârstei lor.

Apelare

- Cu un apel de forma:

? - copil(C).

- obținem, evident, **C = ionut**.
- Pentru:

? - copil(C), write(C), nl, fail.

- vor fi afișați toți copiii găsiți.

Adaugare clauze

- Dacă se ivește un nou copil, și anume, Iulia, trebuie să avem în vedere faptul că ea este cea mai mică. Avem nevoie de un apel de forma:

? – `asserta(copil(iulia))`.

- Putem verifica acum faptul că *Iulia* a fost inserată în baza de date dinamică folosind una din cele două apelări de mai sus, pentru afișarea primului copil, respectiv pentru afișarea tuturor copiilor:

? - `copil(C), write(C), nl, fail`.

Retragere clauze

- Să luăm în continuare cazul în care *Simona* a trecut de vârsta la care mai poate fi numită *copil*. Pentru a o elimina din baza de date, apelăm:

? – `retract(copil(simona))`.

- Din nou putem verifica dacă a fost eliminată, afișând toți copiii:

? - `copil(C), write(C), nl, fail`.

Adaugare clauze

- Să presupunem, în continuare, că toate personajele prezentate mai sus, cât timp au fost copii, au fost buni prieteni.
- Pentru a introduce predicatul *prieteni/2* descris mai sus, chiar de la consola Prologului, folosim următoarea comandă:

? – `assert((prieteni(X,Y):-copil(X), copil(Y))).`

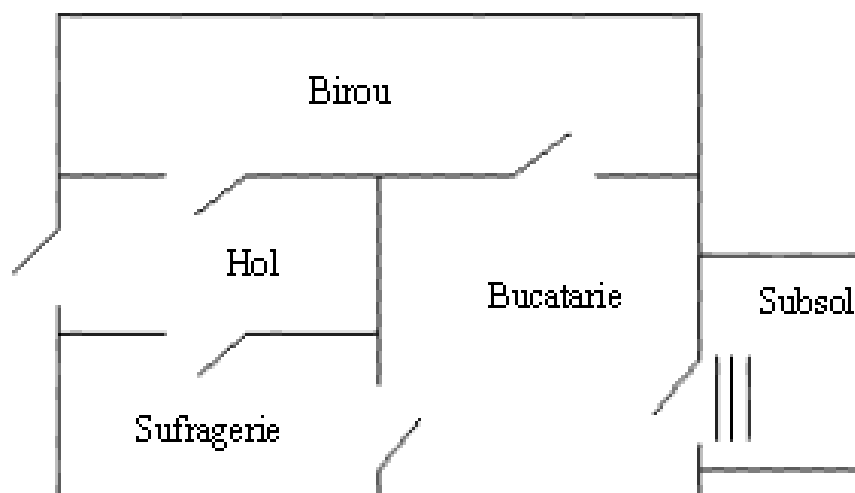
Retragere clauze

- Când toți copiii au trecut de vârsta majoratului, pentru a informa și programul nostru că ei nu mai sunt copii, vom apela:

? – `retractall(copil(_))`.

Exercițiu

- Presupunem că avem următoarea schemă a unei case:



Exercițiu

- Definiți:
 - legăturile dintre camere (eventual printr-un predicat numit *ușă*)



- locul unde se află subiectul (presupunem că inițial se află în birou).



Exercițiu

- La un moment dat, dacă se apelează un predicat numit *mergi/1*, sistemul să spună:
 - care este camera în care se află subiectul
 - dacă nu poate merge în camera specificată ca argument,
 - iar dacă poate, unde poate merge mai departe de acolo.



Exemplu

?- mergi(bucatarie).

Esti in birou

Ai mers in bucatarie

Poti merge in:

birou

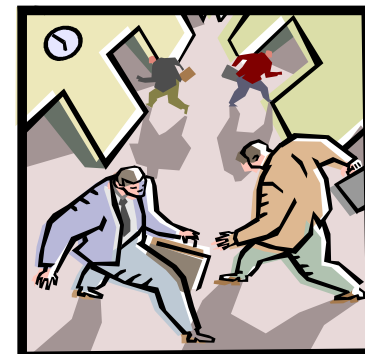
sufragerie

subsol



? – mergi(hol).

Nu poti ajunge direct din bucatarie in hol



Rezolvare

:-dynamic aici/1.

usa(birou, hol).

usa(bucatarie, birou).

usa(hol, sufragerie).

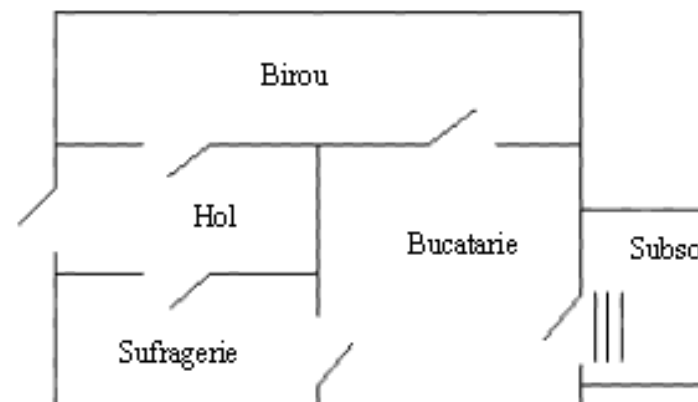
usa(bucatarie, subsol).

usa(sufragerie, bucatarie).

conectat(X,Y) :- usa(X,Y).

conectat(X,Y) :- usa(Y,X).

aici(birou).



Trebuie specificat faptul ca vom defini un predicat in mod dinamic.

Se declara numele predicatului si numarul sau de argumente.

Rezolvare

```
mergi(Camera) :- aici(Aici), not(conectat(Aici,  
    Camera)), write('Esti in '), write(Aici), nl, write('Nu  
    poti ajunge din '), write(Aici), write(' direct in '),  
    write(Camera).
```

```
mergi(Camera) :- aici(Aici), write('Esti in '),  
    write(Aici), nl, write('Ai mers in '), write(Camera),  
    retract(aici(Aici)), assert(aici(Camera)), nl,  
    write('Poti merge in '), nl, unde.
```

```
unde :- aici(Aici), conectat(Aici, Camera), tab(2),  
    write(Camera), nl, fail.
```

```
unde.
```


Rezultate

6 ?- mergi(bucatarie).

Esti in birou

Ai mers in bucatarie

Poti merge in

birou

subsol

sufragerie



Yes

7 ?- mergi(hol).

Esti in bucatarie

Nu poti ajunge din bucatarie direct in hol

Yes

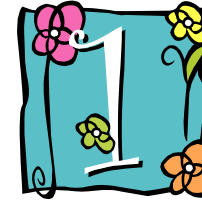


Alt exemplu

- Construiți predicatul *declar/1* care are ca argument o listă de numere întregi și care realizează două alte predicate, *par/1* și *impar/1* care vor avea ca argumente numai numerele care sunt pare, respectiv numai elementele care sunt impare din cadrul listei.



Rezolvare



`:-dynamic par/1.`

`:-dynamic impar/1.`

`declar([]).`

`declar([P|R]) :- P1 is P mod 2, P1 = 0,
assert(par(P)), declar(R).`

`declar([P|R]) :- assert(impar(P)), declar(R).`



Rezultate

39 ?- declar([3, 4, 5, 9, 8, 7]).

Yes

40 ?- par(4).

Yes

41 ?- impar(9).

Yes

42 ?- impar(4).

No

43 ?- par(9).

No

44 ?- impar(X), write(X), tab(3), fail.

3 5 9 7

No

45 ?- par(X), write(X), tab(3), fail.

4 8

No

Functii de memorare

- O functie de memorare salveaza rezultatele unor subcalcule pentru a putea fi folosite direct, in apelari viitoare, fara a mai fi recalulate.
- Prototipul unei astfel de functii este `retin(deductie(arg1, ..., argn))`
- Implementarea se realizeaza astfel:
`retin(deductie(arg1, ..., argn)):-deductie(arg1, ..., argn), asserta(deductie(arg1, ..., argn)).`

Functii de memorare

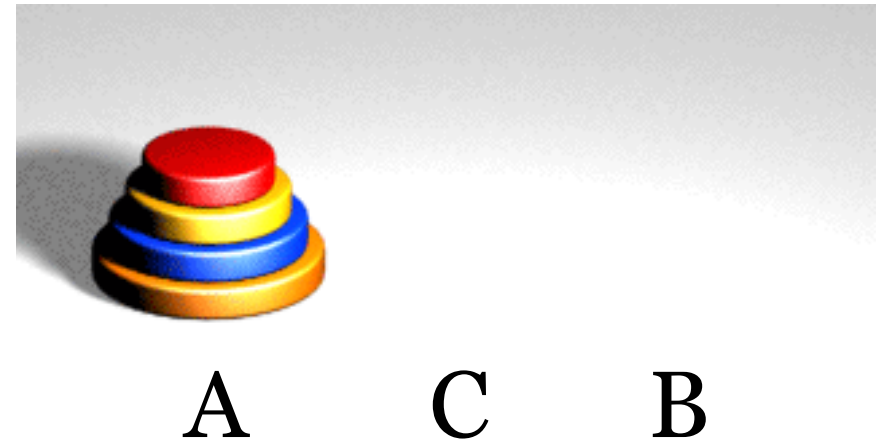
- Se verifica veridicitatea faptului respectiv si, in caz afirmativ, se introduce in memoria Prolog-ului.

`retin(deductie(arg1, ..., argn)):-deductie(arg1, ..., argn), asserta(deductie(arg1, ..., argn)).`

- Data urmatoare cand se doreste atingerea scopului (afirmatia din capul clauzei), solutia va fi gasita direct in memorie, fara a repeta calculele.

Turnurile din Hanoi

- Enunt: Se dau N discuri si 3 cuie: A, B, C.
- Se cere sa se mute toate discurile de pe cuiul A pe cuiul B, folosindu-se drept cui intermediar C.
- Discurile sunt intotdeauna asezate unul peste altul in ordinea crescatoare a marimii.



Turnurile din Hanoi - solutie

- Problema se rezolva recursiv.
- Se muta $N - 1$ discuri de pe cuiul A pe cuiul C.
- Se muta apoi cele $N - 1$ discuri de pe C pe B.
- In final, se aseaza cuiul 1 de pe A pe B.
- Cand avem 1 disc, se muta de pe A pe B.

Turnurile din Hanoi - Prolog

```
hanoi1(N):-hanoi1(N, a, b, c, L), afis(L).
```

```
hanoi1(N,A,B,_C,[(A,B)]):-N = 1.
```

```
hanoi1(N,A,B,C,M):-N > 1, N1 is N - 1,  
    hanoi1(N1,A,C,B,M1),  
    hanoi1(N1,C,B,A,M2),append(M1, [(A, B)|M2],  
    M).
```

```
afis([]).
```

```
afis([X|Rest]):-writeln(X), afis(Rest).
```

Turnurile din Hanoi - functii de memorare

`:-dynamic(hanoi/5).`

`hanoi(N):-hanoi(N, a, b, c, L), afis(L).`

`hanoi(N,A,B,_C,[(A,B)]):-N = 1.`

`hanoi(N,A,B,C,M):-N > 1, N1 is N - 1,`

`retin(hanoi(N1,A,C,B,M1)),`

`hanoi(N1,C,B,A,M2),append(M1, [(A, B)|M2], M).`

`retin(P):-P, asserta(P).`

Turnurile din Hanoi

- Se utilizeaza functiile memo pentru a imbunatati performanta programului initial.
- Solutia acestei probleme, cand sunt N discuri, necesita $2^N - 1$ mutari.
- Modul de solutionare al problemei rezolva in mod repetat subprobleme prin mutarea unui numar identic de discuri.

Turnurile din Hanoi

- O functie de memorare poate fi folosita pentru a retine mutarile facute pentru fiecare subproblema cu un numar mai mic de discuri.
- Cand se ajunge la reapelarea unui fapt care a fost deja calculat, se poate folosi secventa de mutari deja retinuta in memorie, in loc de a le recalcula.
- Se memoreaza prima clauza recursiva, de muta $N - 1$ discuri, a predicatului hanoi si poate fi folosita de a doua apelare recursiva a sa pentru $N - 1$ discuri.

Rulare

```
3 ?- hanoi(4).  
a, c  
a, b  
c, b  
a, c  
b, a  
b, c  
a, c  
a, b  
c, b  
c, a  
b, a  
c, b  
a, c  
a, b  
c, b  
true
```



A

C

B

Predicatul REPEAT



Repeat

- Dacă o anumită clauză trebuie satisfăcută de mai multe ori, facem apel la predicatul *fail* care face ca Prologul să încerce să găsească toate soluțiile (presupunând că nu folosim tăietura).
- Totuși, în unele situații, este necesar să repetăm numai o anumită parte de program, înainte de a continua procesul cu restul clauzei.
- Acest gen de situații apar atunci când vrem să realizăm operații iterative, precum citirea dintr-un fișier ori la realizarea unui meniu.

Repeat

- În general, predicatul *repeat* este folosit în construcții de forma:

```

nume_predicat :- repeat,           % încep bucla
    afisez_meniu,                 % afișez meniul pentru
                                % utilizator
    citesc_optiunea(N),           % citesc data introdusă de
                                % către utilizator
    validez_optiunea(N),         % verific dacă există în
                                % cadrul meniului
    execut_cerinta(N),          % realizez ce era
                                % specificat în
    meniu
                                % pentru N
    verific_oprirea(N),         % verific dacă este
                                % condiția de terminare
    !.                           % oprirea iterației.

```


Exemplu

- Următorul predicat *citesc/o* face citirea de la tastatură a unor valori pe care le afișează apoi utilizatorului.
- Procesul se termină atunci când utilizatorul introduce *gata*.

```
citesc :- repeat, meniu, read(X), write('Am citit '),  
        write(X), write('!'), nl, X == gata, !.
```

```
meniu :- nl, write('-----'), nl, write('Exemplu  
        banal!'), nl, write('Orice introduceti, afisam!'), nl,  
        write('Pentru oprirea repeat-ului, tastati gata. '), nl,  
        write('-----'), nl, nl.
```

Exemplu



1 ?- citesc.

Exemplu banal!

Orice introduceti, afisam!

Pentru oprirea repeat-ului, tastati gata.

| more.

Am citit more!

Exemplu banal!

Orice introduceti, afisam!

Pentru oprirea repeat-ului, tastati gata.

| gata.

Am citit gata!

Yes

Alt exemplu

- Realizați un predicat care să simuleze predicatul predefinit *consult/1*.

Rezolvare

```
compilez(Fisier) :- see(Fisier), repeat,  
    read(X),  
    assert(X),  
    X == end_of_file,  
    !, seen.
```

Rezultate

A screenshot of a Notepad window titled "verif - Notepad". The window has a menu bar with "File", "Edit", "Format", "View", and "Help". The text area contains three lines of Prolog code: "copil(ionut).", "copil(marian).", and "copil(simona).".

```
verif - Notepad
File Edit Format View Help
copil(ionut).
copil(marian).
copil(simona).
```

2 ?- compilez('verif.pl').

Yes

3 ?- copil(X), write_ln(X), fail.

ionut

marian

simona

No

Alt exemplu

- Avem relațiile existente într-o familie date printr-un singur predicat, *persoana(Nume, Sex, Copii)*, unde primul argument reprezintă numele persoanei în cauză, al doilea ne spune dacă este *bărbat* sau *femeie*, iar al treilea este o listă (poate fi vidă) care conține numele copiilor persoanei la care ne referim.

Alt exemplu

- Utilizatorului îi va fi prezentat un meniu care îi permite să facă următoarele operații:
 - Să adauge o persoană (întreabă dacă este bărbat sau femeie); se validează inițial dacă există sau nu în baza de cunoștințe introdusă. Presupunem că nu există două persoane cu același nume.
 - Să șteargă o persoană din baza de cunoștințe.
 - Să adauge informația că X este copilul lui Y.

Alt exemplu

- Continuare meniu:
 - Eliminarea lui X din lista de copii a lui Y.
 - Eliminarea tuturor persoanelor din bază.
 - O opțiune care să-i permită utilizatorului să salveze baza de cunoștințe într-un fișier, respectiv să o citească dintr-un fișier.

Faptele

persoana(andrei, barbat, [cristi, elena]).

persoana(cristi, barbat, [adriana, marius, ovidiu]).

persoana(elena, femeie, [ana]).

persoana(marius, barbat, []).

persoana(ovidiu, barbat, []).

persoana(george, barbat, []).

persoana(adriana, femeie, []).

persoana(ana, femeie, [george]).

Adaugarea unei noi persoane

```
adaugare :- write('Numele celui ce va fi adaugat: '),  
            read(Nume), not(persoana(Nume, _, _)),  
            write('Sexul '), read(Sex),  
            assert(persoana(Nume, Sex, [])), nl,  
            write('Persoana a fost adaugata!').
```

```
adaugare :- write('Exista deja in baza noastra!').
```

Adaugarea unei noi persoane

1 ?- adaugare.

Numele celui ce va fi adaugat: maria.

Sexul femeie.

Persoana a fost adaugata!

Yes

2 ?- persoana(maria, X, Y).

X = femeie

Y = []

Yes

3 ?- adaugare.

Numele celui ce va fi adaugat: andrei.

Exista deja in baza noastra!

Yes

Stergerea unei persoane

```
stergere :- write('Numele celui ce va fi sters: '),  
           read(Nume), persoana(Nume, _, _),  
           retract(persoana(Nume, _, _)), write(Nume),  
           write(' a fost sters!').
```

```
stergere :- nl, write('Nu exista in baza de  
cunostinte!').
```

Stergerea unei persoane

5 ?- persoana(andrei, X, Y).

X = barbat

Y = [cristi, elena]

Yes

6 ?- stergere.

Numele celui ce va fi sters: andrei.

andrei a fost sters!

Yes

7 ?- persoana(andrei, X, Y).

No _

Adaugarea copilului X la Y

```
copil(_X, Y) :- not(persoana(Y, _, _)), write(Y),  
    write(' nu exista in baza noastra de cunostinte!').
```

```
copil(X, Y) :- retract(persoana(Y, SexY, CopiiY)),  
    assert(persoana(Y, SexY, [X|CopiiY])), nl,  
    write('Acum '), write(X), write(' este copilul lui  
' ), write(Y).
```

Adaugarea copilului X la Y

8 ?- persoana(cristi, X, Y).

X = barbat

Y = [adriana, marius, ovidiu]

Yes

9 ?- copil(ion, cristi).

Acum ion este copilul lui cristi

Yes

10 ?- persoana(cristi, X, Y).

X = barbat

Y = [ion, adriana, marius, ovidiu]

Yes

Eliminarea copilului X de la Y

```
elimcopil(_X, Y) :- not(persoana(Y, _, _)),  
    write(Y), write(' nu exista in baza noastra de  
    cunostinte!').
```

```
elimcopil(X, Y) :- persoana(Y, _, Copii),  
    not(member(X, Copii)), write(X), write(' nici nu  
    e copilul lui '), write(Y), write('.').
```


Eliminarea copilului X de la Y

```
elimcopil(X, Y) :- retract(persoana(Y, SexY,  
    CopiiY)), elim(X, CopiiY, CopiiiY),  
    assert(persoana(Y, SexY, CopiiiY)), write('Acum  
' ), write(X), write(' nu mai este copilul lui '),  
    write(Y).
```

```
elim(_, [], []).
```

```
elim(X, [X|R], L) :- elim(X, R, L).
```

```
elim(X, [P|R], [P|L]) :- elim(X, R, L).
```

Eliminarea copilului X de la Y

10 ?- persoana(cristi, X, Y).

X = barbat

Y = [ion, adriana, marius, ovidiu]

Yes

11 ?- elimcopil(adriana,cristi).

Acum adriana nu mai este copilul lui cristi

Yes

12 ?- persoana(cristi, X, Y).

X = barbat

Y = [ion, marius, ovidiu]

Yes

Eliminarea tuturor persoanelor din baza

```
elimintot :- retractall(persoana(_, _, _)), nl,  
             write('Baza de cunostinte este acum goala!').
```

14 ?- elimintot.

Baza de cunostinte este acum goala!

Yes

15 ?- persoana(X, _, _).

No

Salvarea bazei de cunostinte intr-un fisier

```
salvez(Fisier) :- tell(Fisier), salvez, told,  
write('Fisierul a fost salvat').
```

```
salvez :- persoana(Nume, Sex, Copii),  
write('persoana('), write(Nume), write(','),  
write(Sex), write(','), write(Copii), write(').'), nl,  
fail.
```

```
salvez.
```

Salvarea bazei de cunostinte intr-un fisier

2 ?- adaugare.

Numele celui ce va fi adaugat: dan.

Sexul barbat.

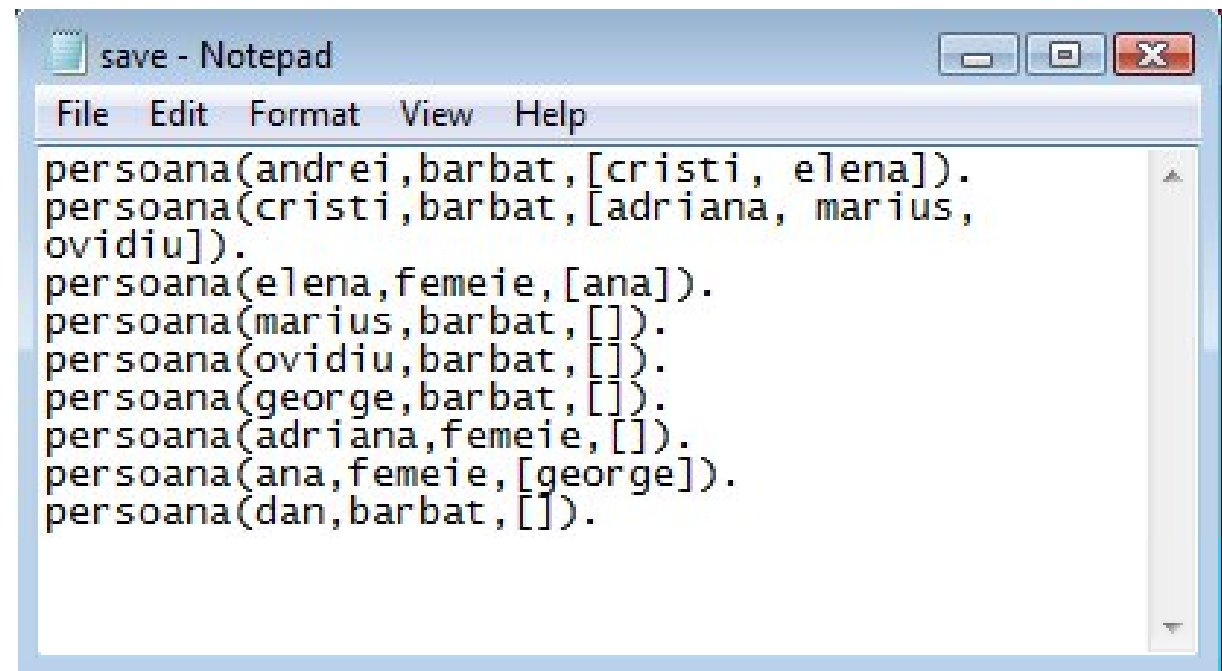
Persoana a fost adaugata!

Yes

3 ?- salvez('save.pl').

Fisierul a fost salvat

Yes



```
File Edit Format View Help
persoana(andrei,barbat,[cristi, elena]).
persoana(cristi,barbat,[adriana, marius,
ovidiu]).
persoana(elena,femeie,[ana]).
persoana(marius,barbat,[]).
persoana(ovidiu,barbat,[]).
persoana(george,barbat,[]).
persoana(adriana,femeie,[]).
persoana(ana,femeie,[george]).
persoana(dan,barbat,[]).
```

Incarcarea unui fisier in memorie

```
deschid(Fisier) :- consult(Fisier), nl,  
    write('Fisierul '), write(Fisier), write(' a fost  
    incarcat. '), nl.
```

```
1 ?- persoana(dan, X, Y).
```

```
No
```

```
2 ?- deschid('save.pl').
```

```
Warning: (c:/users/ruxa & cata/desktop/save.pl:1):
```

```
    Redefined static procedure persoana/3
```

```
% save.pl compiled 0.00 sec, 304 bytes
```

```
Fisierul save.pl a fost incarcat.
```

```
Yes
```

```
3 ?- persoana(dan, X, Y).
```

```
X = barbat
```

```
Y = []
```

Meniul

```
menu :- nl, nl, tab(15), write('Selectati un numar  
din cadrul meniului'), nl, nl, write('1. Listare  
baza de cunostinte. '), nl, write('2. Adaugare  
persoana. '), nl, write('3. Eliminare persoana. '),  
nl, write('4. Adaugarea lui X ca si copil al lui Y. '),  
nl, write('5. Stergerea lui X ca si copil al lui Y. '),  
nl, write('6. Eliminarea tuturor persoanelor. '), nl,  
write('7. Salvarea bazei de cunostinte intr-un  
fisier Prolog. '), nl, write('8. Incarcarea bazei de  
cunostinte dintr-un fisier Prolog. '), nl, write('9.  
Incheiere. '), nl, nl.
```

Partea principala

```
principal :- repeat, meniu, read(Optiune),  
    proceseaza(Optiune), nl, Optiune == 9, !.
```

```
proceseaza(1) :- persoana(Nume, Sex, []),  
    write(Nume), write(' este '), write(Sex), write(' si nu  
    are copii. '), nl, fail.
```

```
proceseaza(1) :- persoana(Nume, barbat, Copii), Copii  
    \= [], write(Nume), write(' este barbat iar copiii lui  
    sunt: '), afiscopii(Copii), nl, fail.
```

```
proceseaza(1) :- persoana(Nume, femeie, Copii), Copii  
    \= [], write(Nume), write(' este femeie iar copiii ei  
    sunt: '), afiscopii(Copii), nl, fail.
```

```
proceseaza(1).
```


Partea principala

proceseaza(2) :- adaugare, !.

proceseaza(3) :- stergere, !.

proceseaza(4) :- write('Copilul X: '), read(X),
write('Parintele Y: '), read(Y), copil(X, Y), !.

proceseaza(5) :- write('Copilul X: '), read(X),
write('Parintele Y: '), read(Y), elimcopil(X, Y), !.

proceseaza(6) :- elimintot, !.

Partea principala

```
proceseaza(7) :- write('Numele fisierului Prolog  
(introduceti-l intre apostrofuri si cu extensia pl):  
'), read(NumeFis), salvez(NumeFis), !.  
proceseaza(8) :- write('Numele fisierului din care  
se face incarcarea (intre apostrofuri si cu  
extensia pl): '), read(Fis), deschid(Fis),!.  
proceseaza(9).
```

Partea principala

afiscopii([]).

afiscopii([P|R]) :- write(P), write(', '), afiscopii(R).

Rulare

3 ?- principal.

Selectati un numar din cadrul meniului

1. Listare baza de cunostinte.
2. Adaugare persoana.
3. Eliminare persoana.
4. Adaugarea lui X ca si copil al lui Y.
5. Stergerea lui X ca si copil al lui Y.
6. Eliminarea tuturor persoanelor.
7. Salvarea bazei de cunostinte intr-un fisier Prolog.
8. Incarcarea bazei de cunostinte dintr-un fisier Prolog.
9. Incheiere.

|: 3.

Numele celui ce va fi sters: cristi.
cristi a fost sters!

Selectati un numar din cadrul meniului

1. Listare baza de cunostinte.
2. Adaugare persoana.
3. Eliminare persoana.
4. Adaugarea lui X ca si copil al lui Y.
5. Stergerea lui X ca si copil al lui Y.
6. Eliminarea tuturor persoanelor.
7. Salvarea bazei de cunostinte intr-un fisier Prolog.
8. Incarcarea bazei de cunostinte dintr-un fisier Prolog.
9. Incheiere.

|: 9.

Yes

Pe saptamana viitoare!

