STRUCTURI DE DATE SPAȚIALE

Reprezentări și operații pe colecții de dreptunghiuri

mat. David D.Dumitru

Universitatea Pitești

Rezumat: Lucrarea tratează câteva aspecte introductive în domeniul Structurilor de Date Spațiale (SDS) și unele reprezentări ale datelor punctuale. În cazul colecțiilor de dreptunghiuri este arătată o reprezentare dinamică și eficientă, împreună cu implementarea în Pascal a unor proceduri care realizează operațiile de bază pe aceste structuri. În final, sunt prezentate aspecte actuale și de perspectivă în domeniu.
Cuvinte cheie: structuri de date, colecții de obiecte, baze de date.

1. Introducere

În domeniul Structurilor de Date (SD) se parcurge un drum lung (teoretic și aplicativ) de la date primitive la SD foarte complexe, care evidențiază, atât relații obiective între componentele acestor structuri, cât și o organizare care permite realizarea unor algoritmi eficienți pentru operațiile de bază sau cele mai frecvent necesare. De ce între aceste tipuri diverse de SD se disting și se impun SDS? Originea răspunsului poate fi găsită în ramurile de bază ale matematicii (geometria, algebra etc.) ca și în raportarea naturală la spațiu a existenței. Astfel, multe din elementele de geometrie plană și în spațiu sugerează tehnici de reprezentare și metode de lucru cu date care în mod natural au o repartiție spațială, geografică așa cum și în existența materială se.întâlnesc un număr de obiecte geometrice (sfere, conuri, piramide etc.). Adesea, un corp poate fi descompus în părți care se încadrează în clasele cunoscute, iar în cazul când are o configurație mai complicată se pot folosi elemente furnizate de teoria curbelor și a suprafețelor pentru a realiza reprezentarea cu ajutorul frontierei.

În sensul prezentei lucrări, prin date spațiale se înțeleg date care reprezintă puncte, linii, figuri geometrice, regiuni, suprafețe și volume. Acestea sunt utilizate în diverse domenii: CAD, CAM, grafică pe calculator, SGBD, robotică, sisteme de informații geografice, recunoașterea formelor, geometrie computațională etc. În acest cadru se disting două clase de probleme:

- reprezentarea datelor și a relațiilor dintre acestea
- optimizarea operațiilor și a aplicațiilor cu astfel de date (timp de execuție cât mai mic și volum de memorie internă/externă minim).

La nivel formal, se disting Tipurile de Date Abstracte (TDA)[1] care sunt cupluri (M,O), alcătuite dintr-un model matematic M și o mulțime de operații O în care se includ generalizări ale operațiilor uzuale (similare rutinelor/procedurilor din limbajele de nivel înalt însoțite de anumite reguli de aplicare). Spre exemplu, dacă $M=Z$ și $O=\{+,-,*,/\}$ avem TDA-mulțimea numerelor întregi. Între TDA clasice se menționează: listele liniare, arborii, grafurile. Operațiile cele mai frecvent folosite sunt: inserția, ștergerea, căutarea unui element. În alocarea dinamică o SD (prin care se înțelege reprezentarea și implementarea într-un limbaj de programare a unui TDA) își începe existența prin a fi vidă apoi, prin inserții succesive, capătă un conținut și dispare o dată cu eliberarea spațiului ocupat în memorie. Operația de căutare este necesară pentru interogările concepute pe SD considerate.

Arborii sunt SD care se folosesc frecvent în reprezentarea datelor spațiale. Alături de date (aspectul formal al informației), în aceste structuri se reprezintă și relațiile spațiale (într-un subarbore se găsesc datele-punctele, părțile dintr-o anumită parte a spațiului). Procesarea arborilor prezintă avantajul recursivității, în acord cu descompunerea recursivă a spațiului până la o rezoluție suficient de fină pentru reprezentarea obiectului spațial pe diverse sectoare (zone, regiuni). Deși se va face referire la cazul obiectelor bidimensionale, este ușoară generalizarea la cazul spațiului n-dimensional (util în baze de date spațiale).

Desigur că există o mare varietate de SDS ilustrate excelent în lucrarea [5] unde este prezentată și o foarte bogată listă de referințe. În practică se aplică totuși puțin importantele rezultate teoretice în domeniu. Metode diverse de reprezentare și de lucru cu colecții de dreptunghiuri care folosesc și alte SD (în special pe bază de arbori) decât cele prezentate aici se găsesc în lucrarea menționată mai sus.

2. Reprezentarea datelor punctuale

Prin Date Punctuale (DP) se înțelege o mulțime de puncte într-un spațiu n-dimensional. În diverse aplicații, DP pot fi disponibile de la început sau pot fi generate pe parcursul execuției, deci dinamic. În repartiția spațială, punctele pot aparține sau nu la o parte (regiune, zonă etc.) a spațiului. Aceste aspecte determină diverse reprezentări ale DP cu aplicabilitate în diverse domenii.

Cea mai simplă reprezentare este dată de Listele Secvențiale (LS) în care operațiile de bază se implementează similar ca în cazul listelor liniare.

O altă metodă este dată de Listele Inverse (LI) în care, pe lângă lista punctelor, se păstrează câte o listă pentru fiecare componentă (coordonată, cheie) a punctului, în fapt o listă de "pointeri" către puncte

O altă metodă este dată de Listele Inverse (LI) în care, pe lângă lista punctelor, se păstrează câte o listă pentru fiecare componentă (coordonată, cheie) a punctului, în fapt o listă de "pointeri" către puncte în care căutarea este rapidă (spre exemplu binară - datorită faptului că listele de "pointeri" se păstrează în ordinea crescătoare a componentelor cărora le sunt asociate).

Metoda Rețelelor Fixe (RF) împarte planul/spațiul în celule (pătrate/cuburi) de mărime egală. Reprezentarea punctelor dintr-o celulă se realizează cu o listă cu legături. Această reprezentare, utilizată de cartografi, prezintă avantajul unui acces rapid la celula dorită și apoi la toate punctele situate spațial în aceasta.

Prin combinarea metodei rețelelor și a arborilor binari de sortare, Finkel și Bentley [2] au inventat q-arborii care în fiecare nod conțin un câmp de informație, două câmpuri cu coordonatele punctului din nod, patru puncte pentru colțurile stânga sus și dreapta jos ale celulei (dreptunghi cu laturile paralele cu axele) în care se situează punctele din subarborele care are ca rădăcină nodul curent și patru pointeri spre fiii corespunzători celor patru celule în care se divizează celula părinte la inserția în aceasta a unui punct. Dacă o dată-punct nu este unică (sunt posibile coliziunile), în fiecare nod se poate adăuga un câmp suplimentar-pointer spre o listă de coliziuni care este memorată în alt loc. Mecanismul logic al inserției este redat de procedura *pqinsert* care urmează unde se presupun următoarele declarații:

```
Type pqarb=^nod;
     nod = record
             x,y:real:{coordonatele punctului}
             p1,p2,p3,p4:pqarb;
     end;
```

Obs. Autorul a omis restul câmpurilor care nu afectează în mod esențial algoritmul și a ignorat coliziunile. Se observă asemănarea cu inserția în cazul arborilor binari de sortare.

Reprezentarea prin această metodă este eficientă pentru operația de căutare, dacă arborele obținut în final este cât mai bine echilibrat. În cazul cînd se cunosc a priori toate punctele de inserat, se sugerează de către Finkel și Bentley[2] o metodă de obținere a unui q-arbore optimizat (cu proprietatea că pentru orice nod din arbore, nici un subarbore din arborele cu rădăcina în nodul considerat să nu conțină mai mult de jumătate din nodurile arborelui). Pentru aceasta, se sortează mulțimea punctelor după prima coordonată, apoi după a doua ș.a.m.d. În această ordine se împart punctele în patru subcolecții ale căror puncte vor fi inserate în cei patru subarbori descendenți ai nodului rădăcină. Analog se procedează cu punctele din fiecare subcolecție.

În cazul când datele de inserat sunt generate în timpul execuției, se poate construi un q-arbore optimizat folosind o metodă propusă de Overmans și van Leeuwen [3] în care arborele este reechilibrat dacă nu îndeplinește condiția arătată mai sus.

```
procedure pqinsert(x1,y1:real; var R:pqarb);
begin
        if r=nil then      begin
                                new(R);
                                with R do  begin
                                                x:=x1;
                                                y:=y1;
                                                p1:=nil;
                                                p2:=nil;
                                                p3:=nil;
                                                p4:=nil;
                                           end
                           end
        else
        if x1<R^.x then                  if y1<R^.y then pqinsert(x1,y1,R^.p3)
                else pqinsert(x1,y1,R^.p1)
        else if y1<R^.y then pqinsert(x1,y1,R^.p4)
                else pqinsert(x1,y1,R^.p2)
end;
```

Obs. În literatură au fost formulate diferite criterii privind arborii echilibrați.

Operația de ștergere a unui nod dintr-un q-arbore este mai complexă dacă arborele va trebui să rămână optimizat. Metoda prin care se reinserează toate nodurile din arborele cu rădăcina în nodul de șters se poate dovedi în multe cazuri costisitoare. Samet [4] propune un procedeu în care un nod (x0,y0) de șters este înlocuit cu un nod (x1,y1) dacă regiunile dintre dreptele x=x0, x=x1, y=y0 și y=y1 nu conțin puncte (sunt vide). Totuși, determinarea nodului (x1,y1) se poate dovedi dificilă. Ideea este de a proceda analog ca la ștergerile în arborii binari, echilibrați de sortare (căutare) unde avem declarațiile și procedura după cum urmează:

```
type Arbs = ^nod;
     nod = record
               elem:tip_elem;
               st,dr:Arbs;
     end
```

Procedura de ştergere apelează funcţia Smin cu codul următor:

```
function Smin(var A:Arb):tip_elem;
begin
if A^.st=nil then begin
                        Smin:=A^.elem;
                        A:=A^.dr;
                  end
else Smin:=Smin(A^.st)
end;
```

Codul procedurii Şterge este dupĺ cum urmează:

```
procedure Şterge(x:tip_elem; var A:Arb);
begin
     if A<>nil then
          if x<A^.elem then Şterge(x,A^.st)
          else if x>A^.elem then Şterge(x,A^.dr)
               else if (A^.st=nil) and (A^.dr=nil) then
                         A:=nil
                    else if A^.st=nil then A:=A^.dr
                         else if A^.dr=nil then A:=A^.st
                              else A^.elem:=Smin(A^.dr)
end;
```

Obs. În locul nodului şters se poate plasa, fie nodul frunză cel mai din dreapta din subarborele stâng, fie nodul frunză cel mai din stânga din subarborele drept, după care nodul frunză este şters

3. Reprezentarea colecţiilor de dreptunghiuri

Pentru a aproxima diverse forme se pot utiliza Colecţii de Dreptunghiuri (CD) din care se selectează o submulţime în care este inclusă forma (figura) dată. În general, se memorează şi frontiera formei (obiectului din spaţiu), dar aceasta este accesată numai dacă se doreşte o precizie mai mare. CD pot fi utilizate în domenii ca: sisteme de informaţii geografice, cartografie (la aproximarea unor forme de relief: lacuri, păduri, dealuri etc.), proiectarea şi verificarea componentelor VLSI pe un cip (spre exemplu la determinarea locului de plasare se verifică anumiţi factori de intersecţie şi separare) etc. De observat că, în cazul ultimului domeniu menţionat, colecţiile implicate pot conţine milioane de componente de dimensiuni foarte mici relativ la spaţiul în care urmează a fi plasate, aspect care obligă la utilizarea unei SDS cât mai performantă în aplicaţii.

3.1. Reprezentarea colecţiilor de obiecte

În alegerea unei reprezentări pentru o colecţie de obiecte din plan sau spaţiu cu o formă arbitrară se impun două probleme: 1). alegerea reprezentării şi 2). organizarea obiectelor care aparţin colecţiilor. În cadrul primei probleme, trebuie decis dacă se alege o reprezentare statică sau una dinamică, ţinând cont de frecvenţa operaţiilor cu obiectele colecţiei şi sistemele concrete pe care se lucrează. Reprezentările obiectelor individuale pot fi grupate în trei categorii principale: a). reprezentări punctuale, b). reprezentări prin părţi caracteristice (în care caz avem variantele: 1. pe baza interiorului obiectului care este descompus în unităţi mai mici-dreptunghiul în pătrate sau o regiune în q-arbori - caracterizate de un singur punct, 2. pe baza frontierei obiectului - un poliedru va fi determinat de feţele sale şi 3. o reprezentare procedurală - o

combinație între variantele 1. și 2. pe baza unui set de reguli bine definite) și c). reprezentări prin partiționarea spațiului în care obiectul este inclus în celule adecvate. Fiecare celulă este asemenea unui buchet care conține referințe la toate obiectele care

```
uses crt;
type mast=^nmast; {pointer spre nodul master}
        punct=^npunct; {pointer spre un nod punct}
        drept=^ndrept; {pointer spreun nod dreptunghi}
        buchet=^nbuchet; {pointer spreun nod buchet}
        nmast= record {nodul master}
                cp:punct; {pointer spre primul nod punct}
                        cd:drept; {pointer spre primul dreptunghi din CD}
                end;
    ndrept=     record {nod dreptunghi}
                        next,prec:drept; {pointer spre nodurile dreptunghi vecine}
                        v1,v2:punct; {pointer spre nodurile dreptunghi}
                end;
    npunct=     record {nod punct}
                        next,prec:punct; {pointer spre punctele vecine din inel}
                        x,y:real; {coord. punctului}
                        b:buchet; {pointer spre primul nod din buchetul dreptunghi
                                asociate punctului}
                end;
    nbuchet=    record {nod buchet}
                        d:drept; {pointer spre dreptunghiul reprezentat din CD}
                        next:buchet; {pointer spre următorul nod din buchet}
        end;
```

îl intersectează [5]. După alegerea unei reprezentări, apare problema implementării acesteia. Se pot folosi liste, arbori, grafuri etc. adaptate la specificul SDS. Pe colecțiile astfel reprezentate și implementate se pot realiza o mare varietate de operații dintre care se menționează: a). operații de bază (inserții, ștergeri ș.a.), b). operații complexe (potriviri exacte sau parțiale, determinarea unor domenii sau frontiere care satisfac anumite condiții, cereri și uniuni de cereri).

3.2. Metodă de reprezentare dinamică a CD

Se pot evidenția diverse metode de reprezentare a CD:de la static la dinamic, de la reprezentări punctuale la structuri complexe care îmbină diverse structuri mai simple. Se menționează că se pot folosi: m1). reprezentări matriceale; m2). reprezentări cu cursoare, utile în limbajele care nu lucrează cu "pointeri" sau pentru salvarea structurii; m3). reprezentări dinamice, cu "pointeri" de genul listelor liniare, multilistelor, grafurilor, arborilor etc.

Reprezentarea care urmează se încadrează în clasa de metode m3 și s-a dovedit foarte eficientă în diverse operații.

Se consideră, mai întâi, cazul unei CD în care dreptunghiurile au laturile paralele cu axele de coordonate. Un dreptunghi va fi definit de două puncte în plan reprezentând, respectiv, colțul din stânga sus și colțul din dreapta jos. Structura, înlănțuită dinamic va conține patru tipuri de noduri:

t1). noduri master cu câmpurile- c1. pointer spre inelul de dreptunghiuri, c2. pointer spre inelul de puncte; t2). noduri dreptunghi cu câmpurile - c1. pointer la următorul nod dreptunghi sau nil dacă nu există, c2. pointer la precedentul nod dreptunghi sau nil când nu există precedent, c3. pointer la nodul punct al colțului din stânga sus, c4. pointer la nodul punct al colțului din dreapta jos; t3). noduri punct cu câmpurile - c1. pointer spre următorul nod punct sau nil când nu există, c2.pointer spre precedentul nod punct sau nil când nu există, c3., c4. conțin coordonatele x,y ale punctului, c5. pointer spre primul nod-buchet, asociat cu punctul din nodul punct curent (punctul este unul din colțurile care determină dreptunghiul punctat de nodul buchet); t4). noduri buchet cu câmpurile - c1. pointer spre nodul dreptunghi, asociat la acel nod punct, c2. pointer spre următorul nod-buchet sau nil când nu mai avem alt dreptunghi asociat (deci care face parte din același buchet). În cazul când există mai multe CD memorate în aceeași structură, vor fi mai multe noduri master, care pot fi înlănțuite în structură de inel. Se prezintă mai jos, în figura 1, un exemplu cu o astfel de structură.

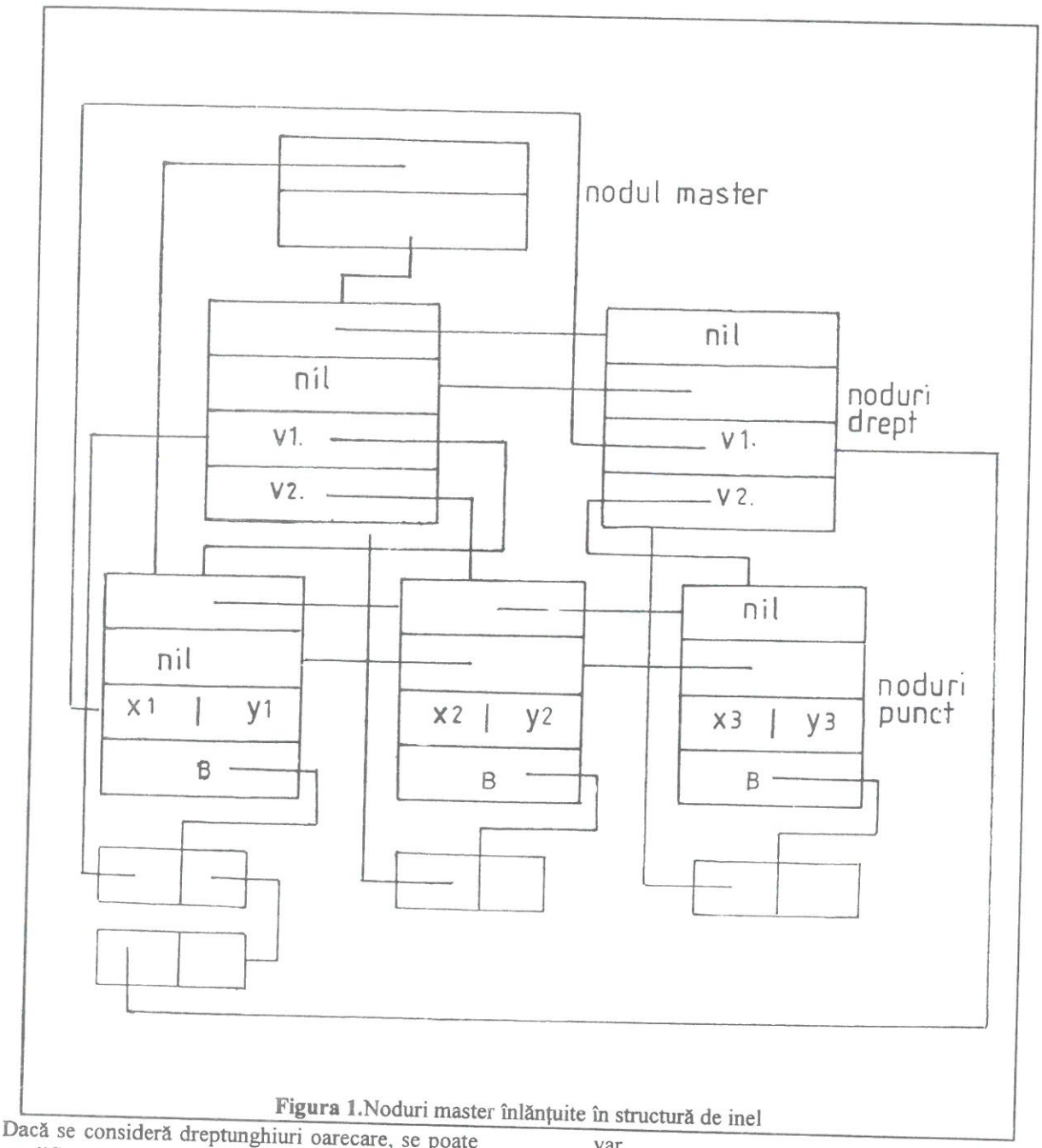

Figura 1. Noduri master înlănţuite în structură de inel

Dacă se consideră dreptunghiuri oarecare, se poate modifica structura în sensul că un dreptunghi să fie determinat de punctul cu coordonatele colţului "cel mai apropiat" de origine, lungimile laturilor şi un unghi dintre axa *ox* şi o latură adiacentă punctului corespunzător colţului care defineşte dreptunghiul (considerată în sensul acelor de ceasornic sau în sens invers).

4. Operaţii pe colecţii de dreptunghiuri

Pentru a prezenta codurile procedurilor referitoare la CD reprezentate ca mai sus, este necesară definirea următoarele tipuri de variabile:

```
var
    op:integer;
    CD:mast;x1,y1,x2,y2:real;
```

{urmează funcţia care inserează în structură un punct de coordonate x1,y1, dacă nu există şi întoarce un pointer spre nodul punct corespunzător}

```
function insertp(cdr:mast;x1,y1:real):punct;
 var
      p:punct;
 begin
      if cdr^.cp=nil then begin
                                new(cdr^.cp);
                                with cdr^.cp^ do
                                     begin
                                       next:=nil;
                                       prec:=nil;
                                       x:=x1;y:=y1;
                                       b:=nil;
                                     end;
                                nsertp:=cdr^.cp;
                              end
       else
       begin
                 p:=cdr^.cp;
                 while (p^.next<>nil) and ((p^.x<>x1) or (p^.y<>y1)) do
          p:=p^.next;
                 if (p^.x=x1) and (p^.y=y1) then insertp:=p
                 else begin
                 new(p^.next);
                 with p^.next^ do begin
                                        next:=nil;
                                        prec:=p;
                                        x:=x1;y:=y1;
                                        b:=nil;
                                      end;
                  nsertp:=p^.next;
                end;
          end;
    end;
```

{urmează procedura de inserție a unui dreptunghi cu colțurile (x1,y1)-stânga sus și (x2,y2)-dreapta jos}

```
    procedure insertd(cdr:mast;x1,y1,x2,y2:real);
    var
      p:drept;
      p1,p2:punct;
```

```
                db:buchet;
begin
 p1:=insertp(cdr,x1,y1);
 p2:=insertp(cdr,x2,y2);
 if cdr^.cd=nil then begin
                    new(cdr^.cd);
                    cdr^.cp:=p1;
                    with cdr^.cd^ do
                      begin
                          next:=nil;
                          prec:=nil;
                          v1:=p1;
                          v2:=p2;
                      end
                          end
else
begin
  p:=cdr^.cd;
  while (p^.next<>nil) and ((p^.v1<>p1) or (p^.v2<>p2)) do
  p:=p^.next;
  if ((p^.v1=p1) and (p^.v2=p2)) then
    writeln('Acest dreptunghi există în CD')
  else begin
       new(p^.next);
       with p^.next^ do begin
                        next:=nil;
                        prec:=p;
                        v1:=p1;
                        v2:=p2;
                      end;
       if p1^.b=nil then begin
                        new(p1^.b);
                        p1^.b^.d:=p^.next;
                        p1^.b^.next:=nil;
                      end
       else begin
            db:=p1^.b;
            while db^.next<>nil do db:=db^.next;
            new(db^.next);
            db^.next^.d:=p^.next;
            db^.next^.next:=nil;
```

```
        end
            end;
        end;
end;

{procedura pentru listarea dreptunghiurilor din colecţie}

procedure listared(cdr:mast);
var
  p:drept;
begin
  p:=cdr^.cd;
  if p=nil then writeln('Colecţia CD este vidă!')
  else begin
                writeln('CD conţine:');
          while p<>nil do begin
                                writeln('v1:',p^.v1^.x:6:2,p^.v1^.y:6:2,
                                    ' v2:',p^.v2^.x:6:2,p^.v2^.y:6:2);
                                  p:=p^.next;
                              end;
              end;
end;

{procedura pentru listarea punctelor din structură}

procedure listarep(cdr:mast);
var
  p:punct;
begin
  p:=cdr^.cp;
  if p=nil then writeln('Colecţia de puncte este vidă!')
  else begin
              writeln('CD conţine punctele:');
              while p<>nil do begin
                          writeln('x:',p^.x:6:2,' y:',p^.y:6:2);
                          p:=p^.next;
                      end;
            end;
end;

{ Programul apelant}
```

```
        Begin
      new(cd);
      cd^.cp:=nil;
      cd^.cd:=nil;
      repeat
       readln;
       clrscr;
       writeln(' Opţiuni posibile:');
       writeln('1:inserţia unui dreptunghi în CD');
       writeln('2:listarea dreptunghiurilor din CD');
       writeln('3:listarea punctelor din CD');
       writeln('10:exit');
       write('Daţi opţiunea:');
       readln(op);
       case op of
        1:begin
            write('Daţi colţul din stânga sus:');
                    readln(x1,y1);
            write('Daţi colţul din dreapta jos:');
                    readln(x2,y2);
            insertd(cd,x1,y1,x2,y2);
          end;
        2:listared(cd);
        3:listarep(cd);
       end;
      until op =10;
End.
```

5. Concluzii şi perspective

Structurile de date spaţiale constituie un domeniu vast de cercetare în informatică. Aplicaţiile de o largă varietate şi o mare importanţă din diverse domenii conferă SDS modernitate şi perspective largi de dezvoltare. Una din aceste perspective constă în utilizarea modelării fuzzy. Mulţimile vagi pot fi reprezentate în SDS. De exemplu, în cazul unei CD se poate considera că apartenenţa unui dreptunghi la CD este dată de o funcţie de apartenenţă, tot aşa cum unele elemente care determină un obiect spaţial pot fi fuzificate.

Bazele de date care sunt atât de necesare în toate domeniile, pe măsura realizării societăţii informaţionale, folosesc tot mai mult SDS pentru a reprezenta informaţiile din societatea contemporană.

Problema abordată în această lucrare ilustrează câteva aspecte ale specificităţii în domeniul vast al SDS.

Bibliografie

1. AHO, A.V., HOPCROFT, J.E., ULLMAN, J.D.: Data Structures and Algorithms, Addison-Wesley, Publishing Company, 1983.
2. FINKEL, R..A., BENTLEY, J.L.: Quad Trees: a Data Structure for Retrieval on Composite Keys. În: Acta Informatica, Vol. 4, No.1, 1974.

3. OVERMARS, M.H., van LEEUWEN, J.: Dynamic Multidimensional Data Structures Based on Quad- and k-d Trees. În: Acta Informatica, Vol. 17, No. 3, 1982.

4. SAMET, H.: Deletion in 2D Quad Tree. În: Communications of the ACM, Vol. 23, No.12, December, 1980.

5. SAMET, H.: The Design and Analysis of Spatial Data Structures, Addison-Wesley Publishing Company, Inc.,1990.

6. TREMBLAY, J.-P., SORENSON, P.G.: An Introduction to Data Structures with Applications, Sec. Edition, McGraw Hill Book Company, 1984.