Structura fizica a bazelor de date relationale

Ioana Ciuciu
ioana.ciuciu@ubbcluj.ro
http://www.cs.ubbcluj.ro/~oana/

Planificare

Sunt posibile mici modificari ale planificarii in timpul semestrului

Saptamana	Curs	Seminar	Laborator
S1	1. Concepte fundamentale ale bazelor de date. Modelare conceptuala	1. Modelul Entitate-Relatie. Modelul relational	1. Modelarea unei BD in modelul ER si implementarea ei in SQL Server
S2	2. Modelul relational de organizare a bazelor de date. Modelare conceptuala		
S3	3. Gestiunea bazelor de date relationale cu limbajul SQL (DDL)	2. Limbajul SQL – definirea si actualizarea datelor	2. Interogari SQL
S4	4. Gestiunea bazelor de date relationale cu limbajul SQL (DML)		
S5-6	5-6. Dependente functionale, forme normale	3. Limbajul SQL – regasirea datelor	3. Interogari SQL avansate
S7	7. Interogarea bazelor de date relationale cu operatori din algebra relationala	4. Proceduri stocate	4. Proceduri stocate. View. Trigger
S8	8. Structura fizica a bazelor de date relationale		
S9	9. FARA CURS (30 nov.)	5. View-uri. Functii definite de utilizator. Trigger	
S10-11	10-11. Indecsi. Arbori B. Fisiere cu acces direct	6. Formele normale ale unei relatii. Indecsi	
S12	12. Evaluarea interogarilor in bazele de date relationale		
S13	13. Extensii ale modelului relational si baze de date NoSQL	7. Probleme	Examen practic
S14	14. Aplicatii		

Planul cursului

- Curs 8
  - Structura fisierelor
    - Probleme ce trebuie rezolvate
  - Tipuri de interogari
  - Cautarea secventiala
  - Cautarea in colectii de date ordonate dupa valorile unei chei
- Curs 9 : NU SE TINE (30 nov.)
- Curs 10-11
  - 2,3-arbore
  - B-arbore
  - Organizarea directa
  - Index pentru un atribut oarecare
  - Gestiunea indexurilor

Structura fisierelor

Tipuri de memorie:

- **internă**: nepersistentă
- **externă**:
  - persistentă
  - acces secvenţial şi direct (disc magnetic, memorie adresabilă), numai acces secvenţial (benzi magnetice)

Caracteristici ale memoriei externe:

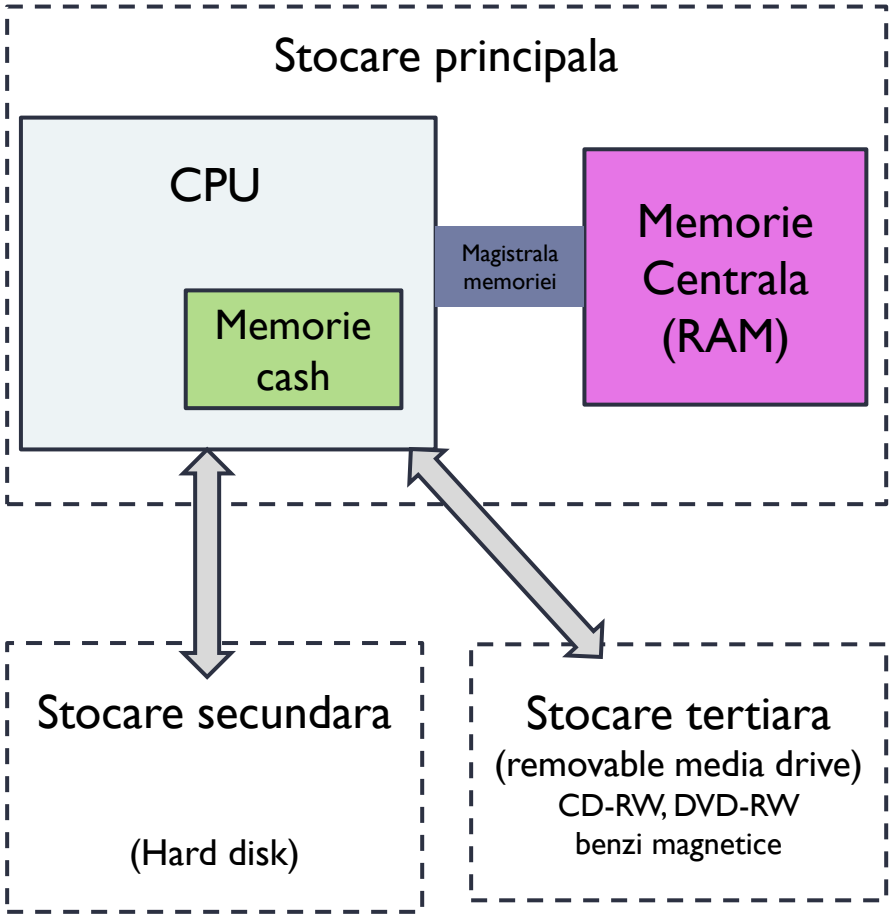
- **capacitatea de stocare**
- **timp de acces**: "timpul ce se consumă din momentul emiterii unei cereri de citire sau scriere și până în momentul când începe efectiv transferul de date"
- **rata de transfer**: "cantitatea de informație ce se transferă într-o secundă"
- **dimensiunea unui bloc** (înregistrare fizică). Blocul este unitatea de transfer dintre memoria internă şi cea externă.

Bază de date:

- **nivel logic** (în aplicaţii): conform unui model de organizare
- **nivel fizic**: colecţie de fişiere
- un fişier este
  - o colecţie de înregistrări logice (de exemplu, pot corespunde la liniile unui tabel din modelul relaţional dacă tabelul se memorează într-un fişier)
  - o colecţie de blocuri pe suport (succesiune de octeţi)

Structura fisierelor

- Stocarea datelor (la nivel digital):
  - Stocare primara
  - Stocare secundara
  - Stocare tertiara

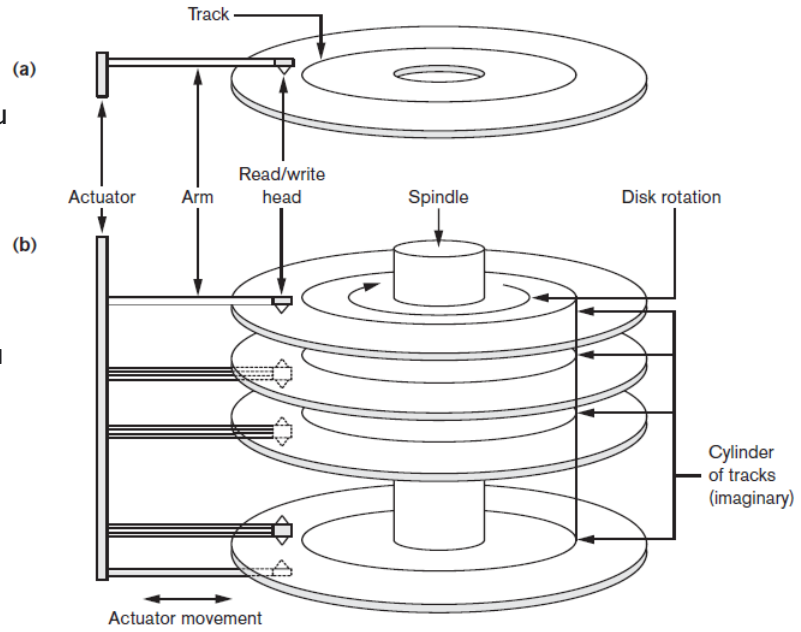


Structura fisierelor

- Descrierea hardware a discurilor (*disk devices*)

(a) disc cu o singură față cu hardware de citire/scriere

(b) Un pachet de discuri cu hardware de citire/scriere



Sursa: [EI10] ELMASRI R., NAVATHE S.B., Fundamentals of Database Systems, 6th Ed., Pearson, 2010.

Structura fisierelor

- Descrierea hardware a discurilor (*disk devices*)

- Gruparea **bitilor** in **bytes** (in general, *8 biti*), 1 byte = 1 octet (ex., 'h', '9', '$')
- **Capacitatea** discului: numarul de bytes pe care il poate stoca

- kilobyte (kB): 1000 bytes
- megabyte (MB): 1000^2
- gigabyte (GB): 1000^3
- terabyte (TB): 1000^4 bytes
- petabyte (PB): 1000^5
- exabyte (EB): 1000^6
- zettabyte (ZB): 1000^7
- yottabyte (YB): 1000^8
- ronnabyte (RB): 1000^9
- ...

Structura fisierelor

Probleme privind gestiunea fişierelor:

1. **Cum se realizează ştergerea unei înregistrări?**
   - pe locul ocupat se mută înregistrări
   - marcarea zonelor şterse şi păstrarea unei liste a lor. La următoarea operaţie de adăugare se pot folosi înregistrări din această listă (dacă lista nu e vidă).
2. **Gruparea înregistrărilor în blocuri**
   - schimbul dintre memoria internă şi suport se face prin **blocuri**
   - un bloc poate conţine mai multe înregistrări (toate sunt transferate simultan)
   - dacă o înregistrare se memorează în mai multe blocuri, atunci poate creşte timpul de acces
3. **Memorarea înregistrărilor de lungime variabilă** (provenite din memorarea coloanelor de lungime variabilă)
   - fiecare înregistrare se memorează pe spaţiul de memorie necesar (există multe metode de regăsire a valorilor coloanelor) - apar probleme la acces şi modificare
   - se rezervă o zonă de memorie maximă pentru a se putea memora orice înregistrare
   - formatarea spaţiului în zone de lungime fixă şi memorarea unei înregistrări într-o mulţime de astfel de zone (consecutive, înlănţuite)
   - formatarea spaţiului în două zone:
     - o zonă secundară unde se memorează valorile coloanelor de lungime variabilă
     - o zonă principală care conţine valorile coloanelor de lungime fixă şi adresele valorilor (eventual şi lungimea valorilor) pentru coloanele de lungime variabilă. Fiecare înregistrare din zona principală va avea o lungime fixă.

Structura fisierelor/Tipuri de interogari

4. Gestiunea zonelor tampon (zonă din memoria internă unde se poate memora un bloc)

- dacă o înregistrare necesară se află în memoria internă (într-o zonă tampon), atunci se micşorează timpul de acces (nu mai trebuie transferată de pe suport)
- modificarea unei înregistrări se face în memoria internă. Modificările făcute se trec în baza de date la anumite momente.
- sunt necesari algoritmi care caută o zonă tampon în care se va citi un nou bloc. Dacă zona tampon care se înlocuieşte conţine înregistrări modificate, atunci aceasta trebuie rescrisă pe suport înainte de a se citi alte date. **Probleme ce pot să apară**.

Tipuri de interogări într-un tabel memorat sau tabel temporar: se cer înregistrările pentru care o condiţie are valoarea logică de ***true***

- **Condiţie elementară** cu un atribut de căutare (atribut cheie sau atribut oarecare)
  - **atribut** *operator_relaţional* valoare
  - **expresie** *operator_relaţional* valoare
  - **expresie** $\in [v_1, v_2]$
  - **expresie** $\in \{v_1, v_2, ..., v_n\}$
  - **expresie** *LIKE* **şablon**
  - se defineşte o distanţă ***d*** peste valorile atributului şi se cer înregistrări pentru care valoarea atributului:
    - d(**atribut**, v0) <= d0
    - cea mai apropiată valoare de v0,
    - cele mai apropiate **n** valori de v0
- **condiţie compusă** cu operatorii logici: **not, and, or**

Tipuri de interogari

Pentru fiecare tip de interogare trebuie să existe cel puţin un algoritm care determină răspunsul.

Parametri importanţi la determinarea răspunsului unei interogări (care depind de structura fizică):

- **timpul (mediu) de răspuns** pentru o mulţime de interogări
- **spaţiul suplimentar de memorie folosit** la memorarea sursei de date

Accesul la înregistrări
Pentru a regăsi o înregistrare trebuie ca ea să fie localizată pe suport. Această localizare (accesul la înregistrare) se poate face:

- **secvenţial**: localizarea se face prin parcurgerea înregistrărilor care o preced pe suport
- **direct**: localizarea se face prin intermediul unei adrese calculate (adresa calculată poate fi folosită pentru a preciza de unde se începe căutarea înregistrării)
- **indirect**: adresa înregistrării este memorată împreună cu alte date (în altă zonă de memorie, într-un index)

Organizarea fişierelor: stabilirea unei relaţii între înregistrările fişierului şi poziţia lor pe suport

- Fie $F=\{r_1, r_2, ..., r_n\}$ colecţia de înregistrări dintr-un fişier (privit la nivel logic, ca liniile unui tabel/relaţii). In acest fişier trebuie efectuate operaţiile următoare: ***adăugarea, ştergerea, modificarea valorilor, căutarea, parcurgerea (totală sau parţială).***
- Există multe metode de organizare a fişierelor, care permit micşorarea timpului de răspuns pentru anumite tipuri de operaţii.

Cautarea secventiala

Determinarea răspunsului la diverse tipuri de interogări

1. Căutare secvenţială pentru interogarea C = K0, unde C este o cheie

Deoarece C este cheie, răspunsul poate fi:

- o singură înregistrare ("*căutare cu succes*")
- mulţimea vidă ("*căutare fără succes*")

După descrierea algoritmilor de căutare este necesară **complexitatea** lor (pentru a putea face o comparaţie între ei). Din această cauză se poate presupune că toată colecţia (tot fişierul) este în memoria internă. Algoritmii se pot uşor modifica pentru a se citi înregistrările dintr-un fişier.

Fie: K_i = valoarea cheii **C** pentru înregistrarea $\mathbf{r_i}$, deci $K_i = \Pi_C(r_i)$, i=1, ..., n.

Algoritm care determină răspunsul la interogarea amintită: se compară K0 cu K[1], K[2], ...

Cautarea secventiala

Pentru **complexitatea algoritmului** se poate lua **numărul de comparaţii efectuate între chei**.

Observaţie. Căutare secvenţială după **valoarea unui atribut** cere consultarea întregului fişier.

Probleme asemănătoare:

- ordinea de memorare a înregistrărilor de lungime variabilă, pentru o căutare secvenţială, dacă o înregistrare r_i, i=1,...,n, are **lungimea** L_i şi **probabilitatea** de a fi căutată p_i.
- ordinea de testare a condiţiilor C_i din condiţia compusă C_1 **and** C_2 **and ... and** C_n, dacă C_i necesită un timp T_i pentru evaluare şi are probabilitatea P_i de a fi adevărată.

Cautarea in colectii de date ordonate

2. Căutarea în colecţii de date ordonate pentru interogarea C = K0, unde C este o cheie

Deoarece C este cheie, toate valorile acesteia sunt distincte: $K_i \neq K_j, i \neq j$.

Presupunem că înregistrările sunt ordonate după valorile atributului C, deci:

$$K_1 < K_2 < ... < K_n .$$

Pentru acest tip de colecţie, cu algoritmul de **căutare secvenţială** se micşorează numărul de comparaţii pentru "căutarea fără succes".

Algortimul de căutare binară (logaritmică, bisectare)

Cautarea in colectii de date ordonate

Pentru a determina **complexitatea algoritmului de căutare binară**, se construieşte un arbore binar cu **n** noduri interioare (în desen aceste noduri apar încercuite) şi (n+1) noduri terminale (încadrate):

- dacă n=0, atunci arborele este: 0
- dacă n≠0, atunci nodul rădăcină este $\lceil n/2 \rceil$, subarborele stâng este arborele binar cu $\lceil n/2 \rceil$-1 noduri, iar subarborele drept este arborele binar cu $\lfloor n/2 \rfloor$ noduri la care etichetele se măresc cu $\lceil n/2 \rceil$. Cei doi subarbori se construiesc după acelaşi procedeu.

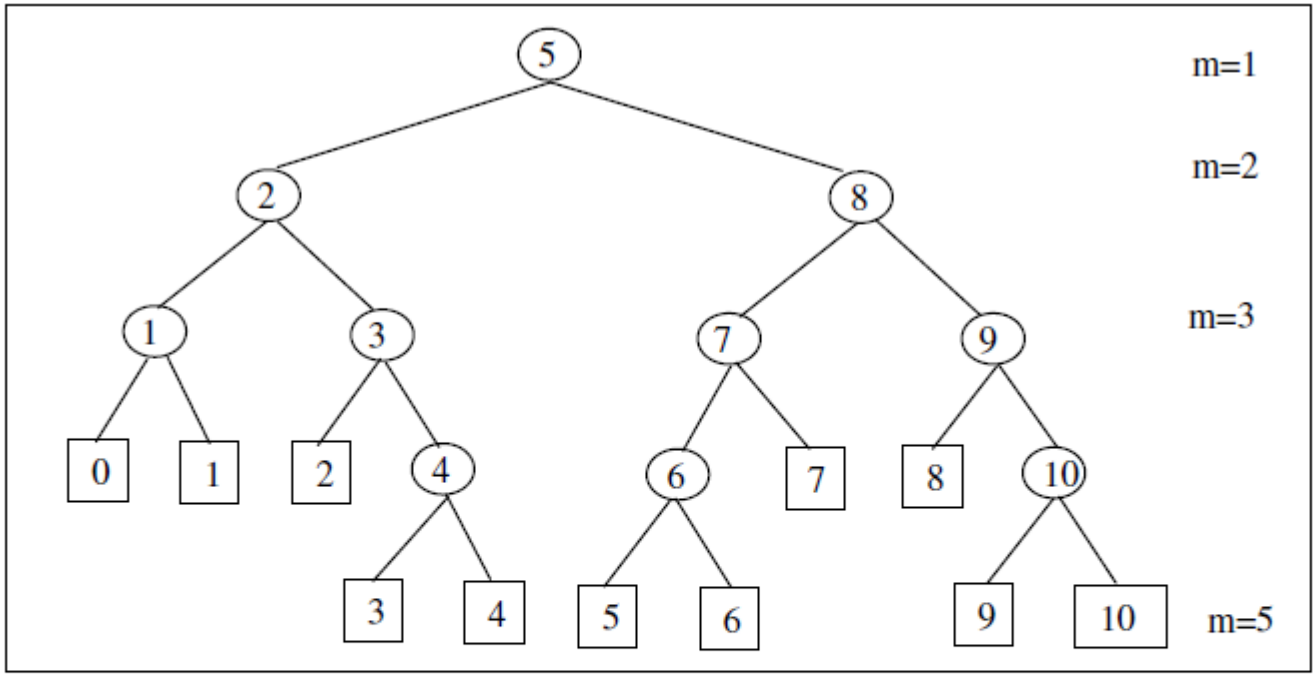
Arborele binar care se construieşte va avea nodurile interioare corespunzătoare valorilor **i** (poziţia în vector) din comparaţiile **K[i]=K0** care se fac în algoritm.

Algoritmul de căutare binară se poate termina:

- într-un nod interior (căutare cu succes)
- într-un nod terminal (căutare fără succes)

De exemplu, pentru **n=10** se obţine arborele din figura următoare.

Cautarea in colectii de date ordonate



Dacă datele nu încap pe **m-1** nivele $\Rightarrow$ n > $2^0+2^1+...+2^{m-2}=2^{m-1}-1$, dar încap pe **m** nivele:

$$2^{m-1} \le n < 2^m,$$

atunci arborele binar are nodurile interioare situate pe **m** nivele (pentru figură, numărul de nivele este 4), iar nodurile terminale pe cel mult două nivele. Din relaţia precedentă se obţine:

$$m-1 \le \log_2(n) < m \Rightarrow m-1 = \lfloor \log_2(n) \rfloor \Rightarrow m = \lfloor \log_2(n) \rfloor + 1$$

Cautarea in colectii de date ordonate

Observaţie. Algoritmul de căutare binară necesită:

- **minimum 1** comparaţie
- **maximum** $\left\lfloor \log_2(n) \right\rfloor + 1$ comparaţii
- un **număr mediu de comparaţii**: $O(\log n)$

Comparaţie între căutarea secvenţială şi căutarea în colecţii ordonate.

Alţi algoritmi de căutare în colecţii ordonate:

- **căutarea Fibonacci**: indicii folosiţi pentru comparaţii sunt calculaţi folosind numerele lui Fibonacci
- **căutarea prin interpolare**: indicii folosiţi pentru comparaţii sunt calculaţi folosind o interpolare liniară

Cautarea in colectii de date ordonate

Memorarea colecţiei de date ca un arbore binar

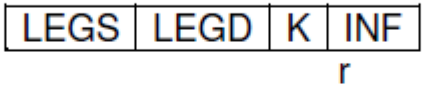
Algoritmul de căutare binară în colecţii ordonate:

- foarte rapid
- greu de respectat condiţia cerută: **înregistrările sunt ordonate** (mai ales pentru colecţii dinamice)

Soluţie: **memorarea colecţiei ca un arbore binare:**

- fiecare înregistrare se memorează într-un nod al arborelui binar
- **subarborele stâng** pentru un nod cu valoarea **v** pentru cheie va conţine înregistrări cu valori mai mici decât **v** pentru cheie
- **subarborele drept** pentru un nod cu valoarea **v** pentru cheie va conţine înregistrări cu valori mai mari decât **v** pentru cheie

Structura unui nod (valorile din înregistrare se memorează în K şi INF):

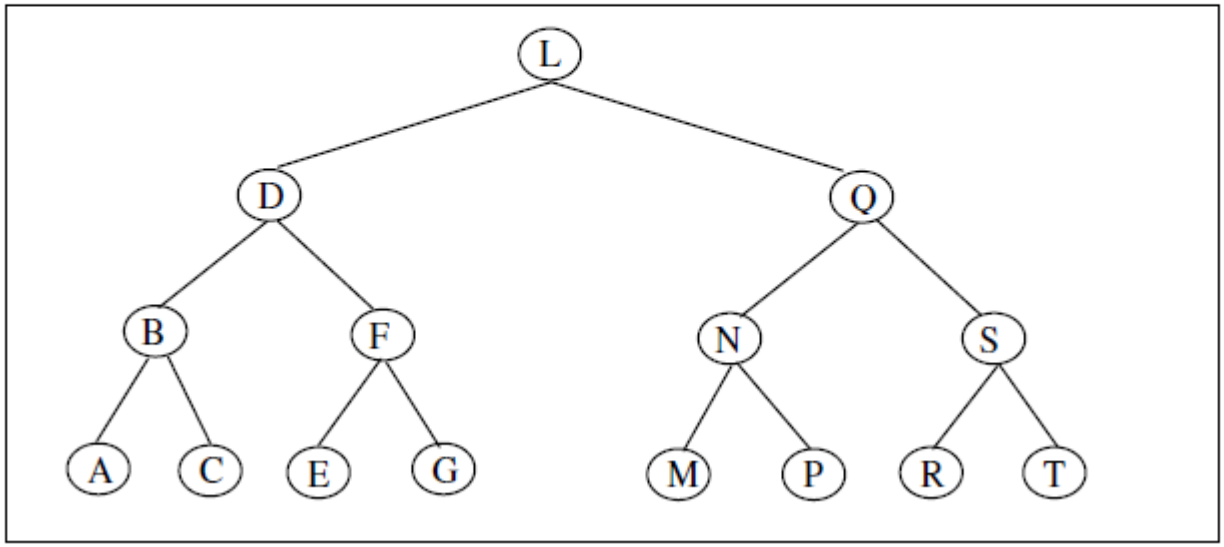


Operaţii necesare pentru acest arbore binar:

- **căutarea** unei înregistrări cu valoarea **K0** pentru cheie - **algoritm**
- **adăugarea** unei înregistrări - **pe exemple**
- **ştergerea** unei înregistrări
- **parcurgerea arborelui** (parţial: între două valori, total)

Cautarea in colectii de date ordonate

In continuare se vor da arborii binari obţinuţi **după adăugarea unor date** (se pun în evidenţă numai valorile pentru cheie). Considerăm că valorile pentru cheie sunt din mulţimea următoare: {**L, D, B, Q, N, F, S, R, T, M, E, G, P, A, C**}, iar adăugarea în arbore se face în ordinea în care valorile se precizează în mulţime. Arborele obţinut după ce s-au efectuat toate operaţiile de adăugare este cel din figură.

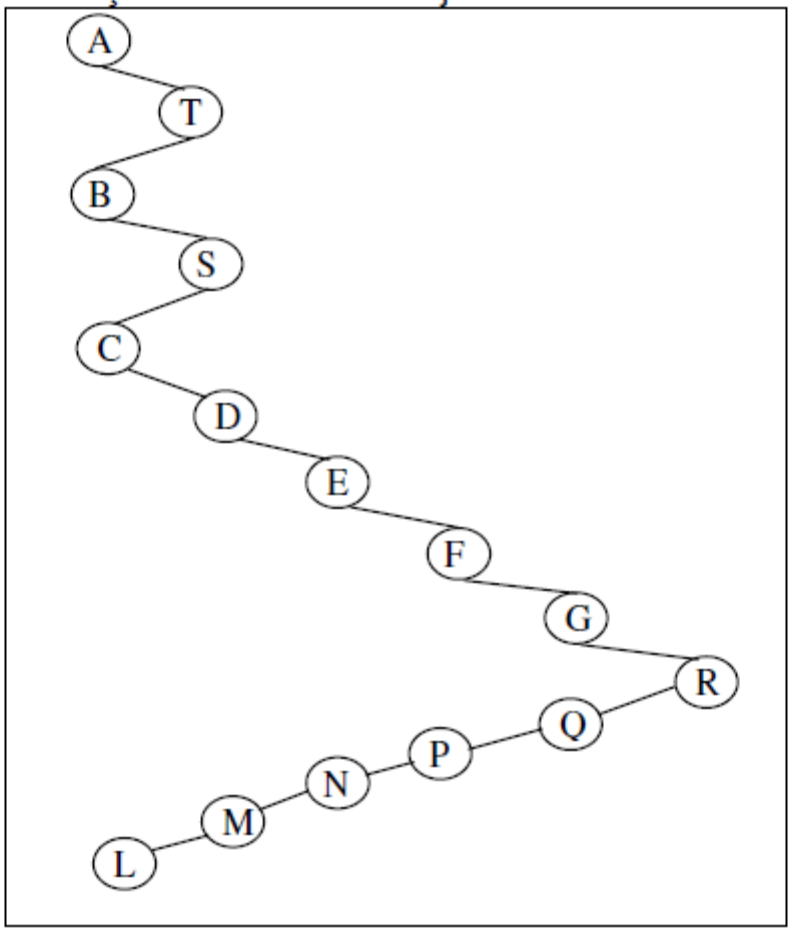


Cautarea in colectii de date ordonate

Dacă valorile pentru cheie sunt aceleaşi, dar furnizate în altă ordine: {A, T, B, S, C, D, E, F, G, R, Q, P, N, M, L}, atunci se obţine arborele de mai jos.

Cautarea in colectii de date ordonate

Dacă valorile pentru cheie sunt aceleaşi, dar furnizate în altă ordine: {A, T, B, S, C, D, E, F, G, R, Q, P, N, M, L}, atunci se obţine arborele de mai jos.



Cautarea in colectii de date ordonate

Observaţii:

- Arborele din ultima figură este "degenerat", căutarea în el se reduce la o căutare secvenţială.
- Forma arborelui, deci şi timpul de căutare în aceşti arbori, depinde de ordinea în care datele s-au adăugat la colecţie. Timpul minim se obţine pentru un **arbore optimal** (în care terminalele sunt memorate pe cel mult două nivele consecutive), asemănător cu arborele construit pentru căutarea binară.

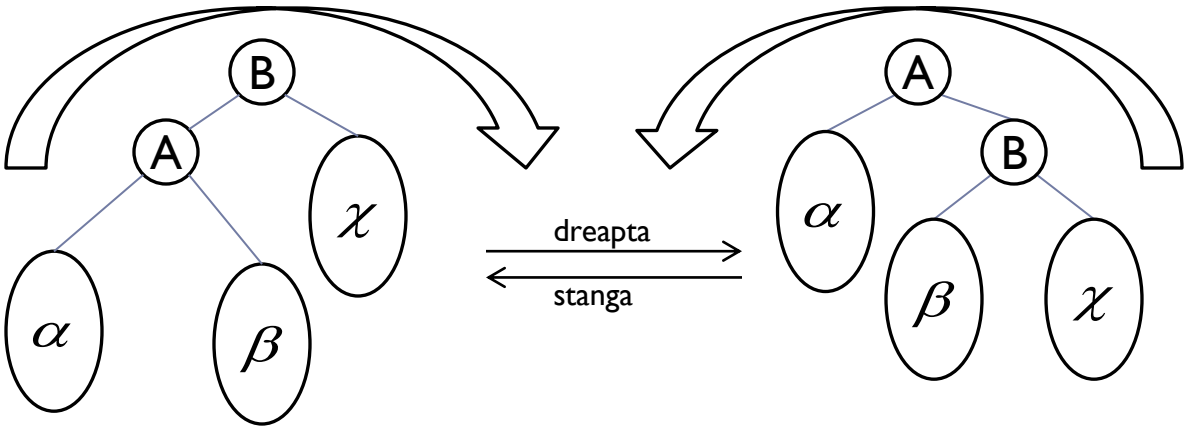
Definiţii.

1. **Înălţimea unui arbore** este lungimea celui mai lung drum de la rădăcină la terminale (lungimea unui drum este dat de numărul de vârfuri).
2. Un **arbore binar este echilibrat** dacă pentru fiecare nod diferenţa dintre înălţimea celor doi subarbori este 0, 1 sau -1.

Observaţie. Dacă prin operaţii de adăugare sau ştergere un arbore echilibrat se transformă în unul care nu mai este echilibrat, atunci printr-un număr mic de transformări el poate să fie echilibrat.

Cautarea in colectii de date ordonate

- Rotirea unui arbore
  - Conditii:
    - Ordinea nodurilor terminale nu se schimba
    - Sub-arborele drept e mai mare decat radacina, iar sub-arborele stang e mai mic decat radacina



Cautarea in colectii de date ordonate

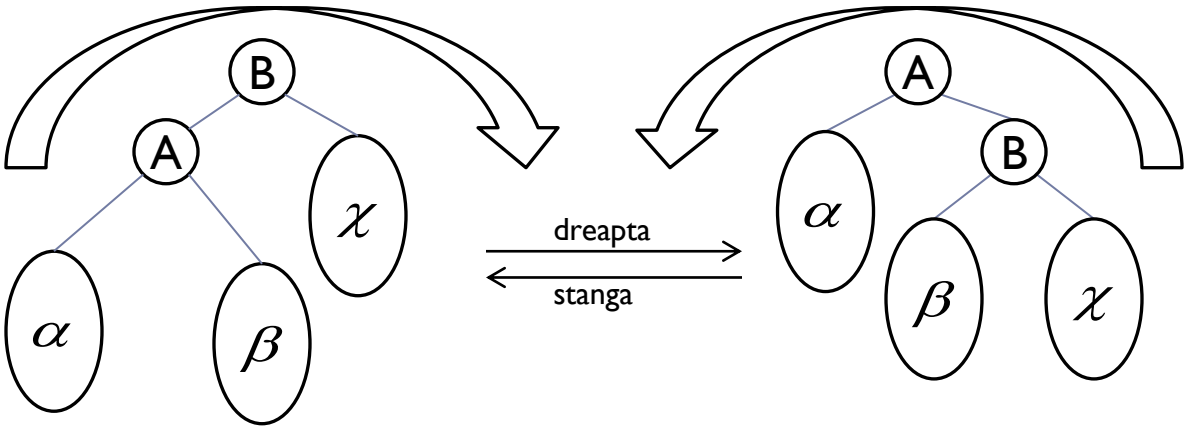
- Rotirea unui arbore
  - Algoritm:

Rotirea spre **dreapta:**

```
Pivot = Root.S
Root.S = Pivot.D
Pivot.D = Root
Root = Pivot
```

Rotirea spre **stanga**:

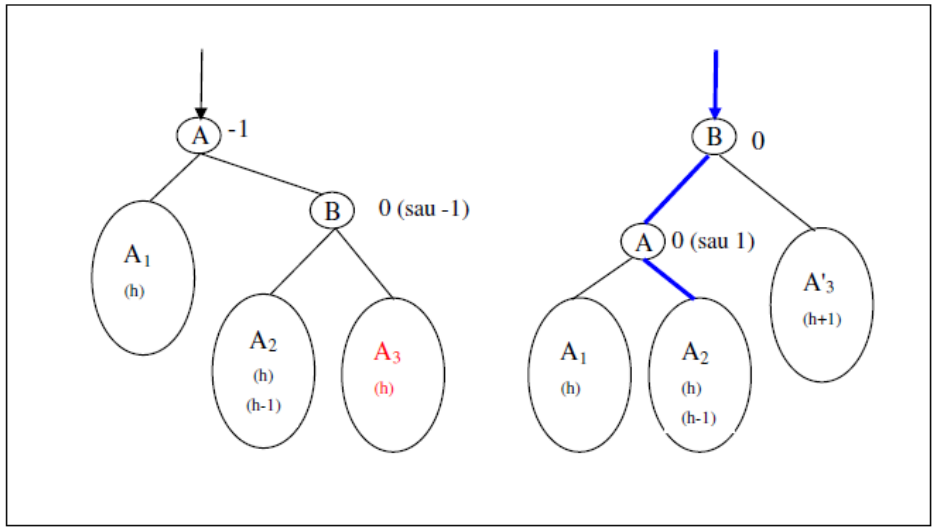
```
Pivot = Root.D
Root.D = Pivot.S
Pivot.S = Root
Root = Pivot
```



Cautarea in colectii de date ordonate

Exemplu. La arborele A_3 din figura următoare se adaugă o valoare care modifică înălţimea (se măreşte cu 1). Inălţimea subarborilor este trecută între paranteze, iar la nodurile A şi B apare diferenţa dintre înălţimea celor doi subarbori. In nodul A se micşorează diferenţa dintre înălţimea celor doi subarbori, deci după adăugare arborele nu va mai fi echilibrat. Prin transformarea ce apare în partea dreaptă a figurii se face o echilibrare pentru arborele cu rădăcina în A. Această echilibrare (transformare) necesită puţine atribuiri în zona de memorie unde se memorează arborele (LEGD pentru A, LEGS pentru B, legătura care face referire la A se schimba la B). Alte modificări pot să intervină pe drumul din rădăcină la B.

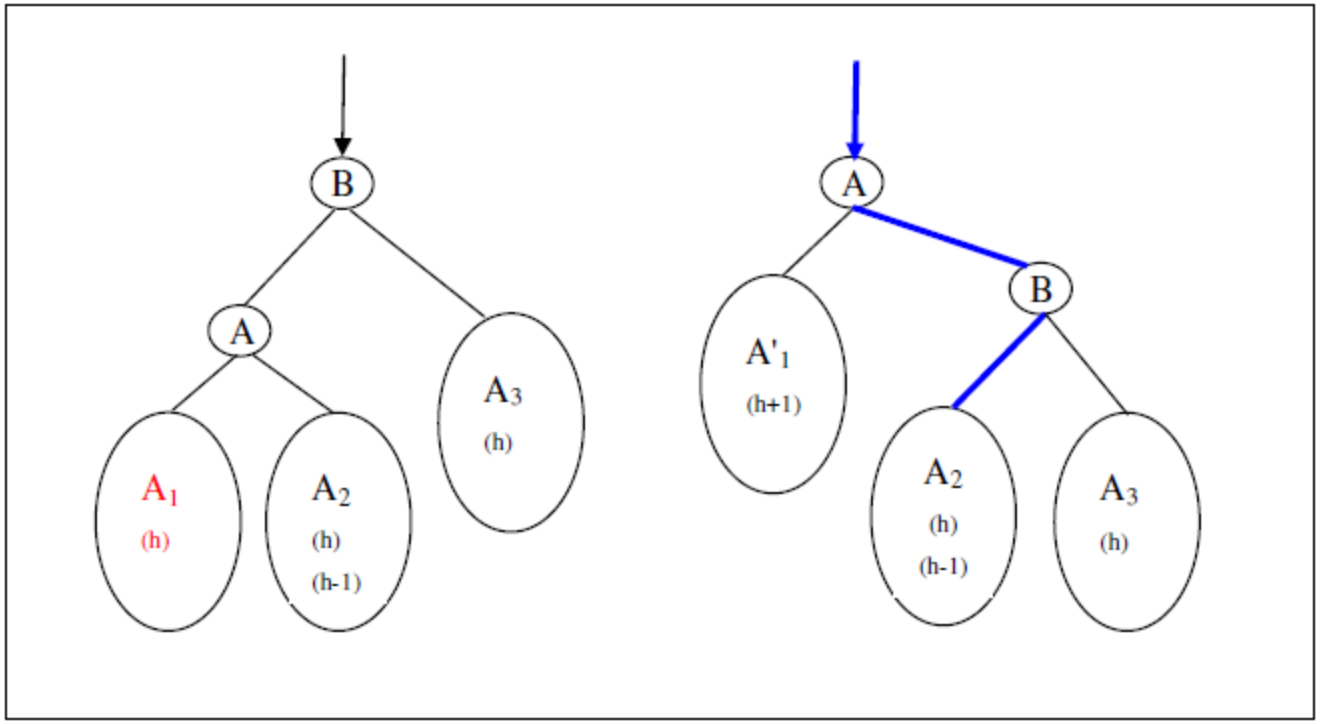
Varianta 1:



Precizare. In [Kn76] sunt enumerate toate cele **şase** transformări existente pentru echilibrare. S-a marcat cu culoarea roşie subarborele la care se adaugă un nod şi la care se măreşte înălţimea.

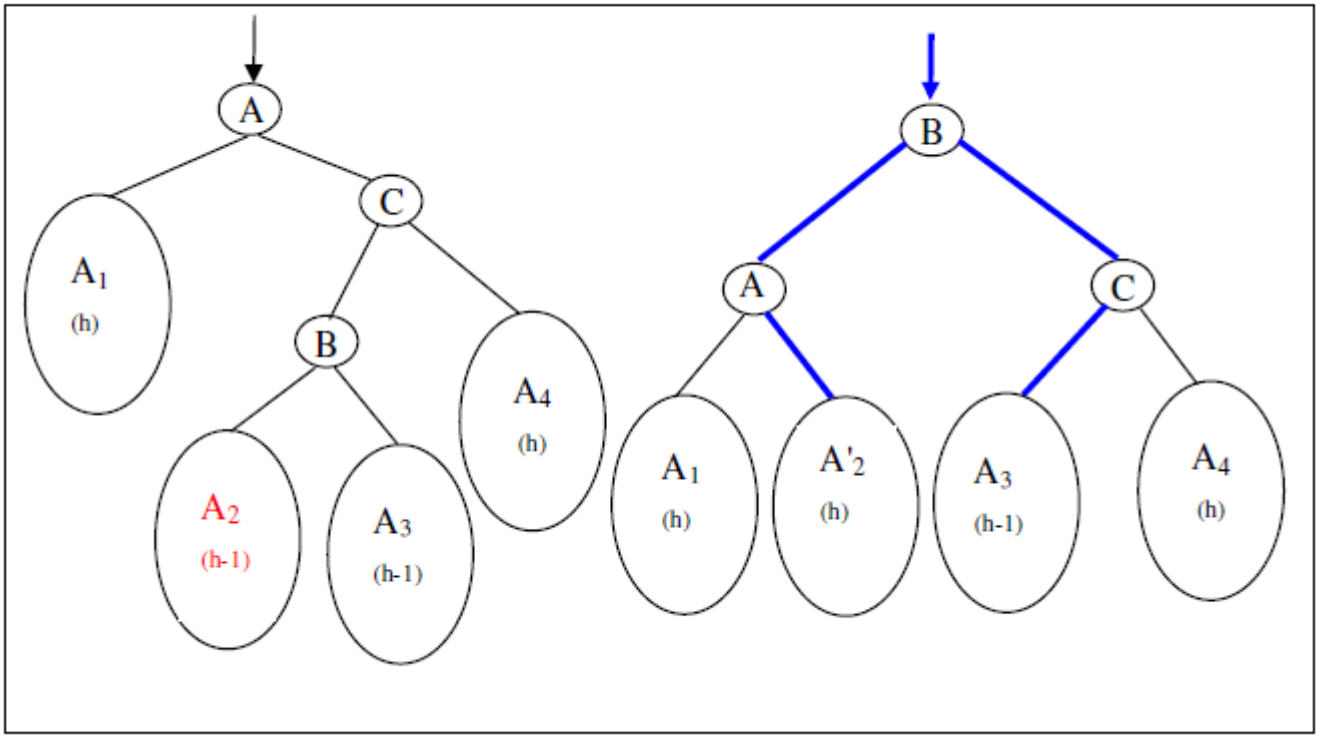
Cautarea in colectii de date ordonate

Varianta 2:



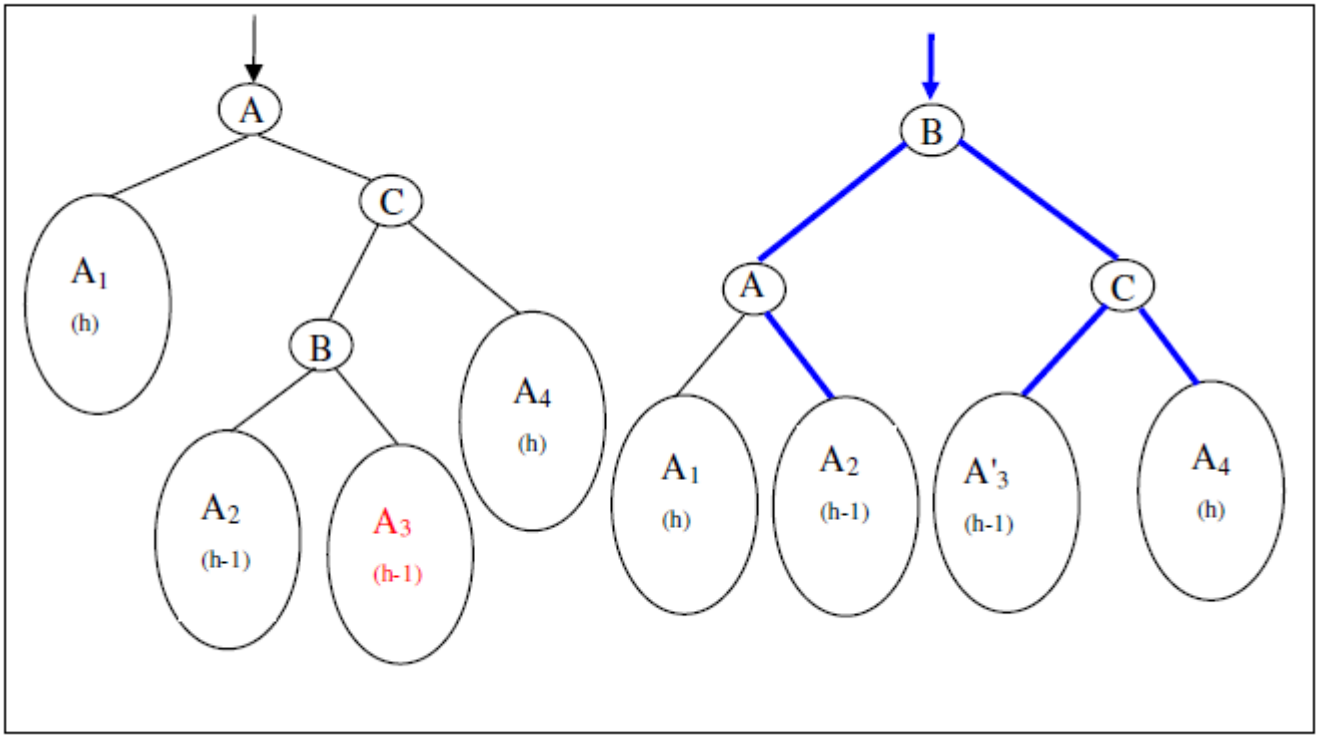
Cautarea in colectii de date ordonate

Varianta 3:



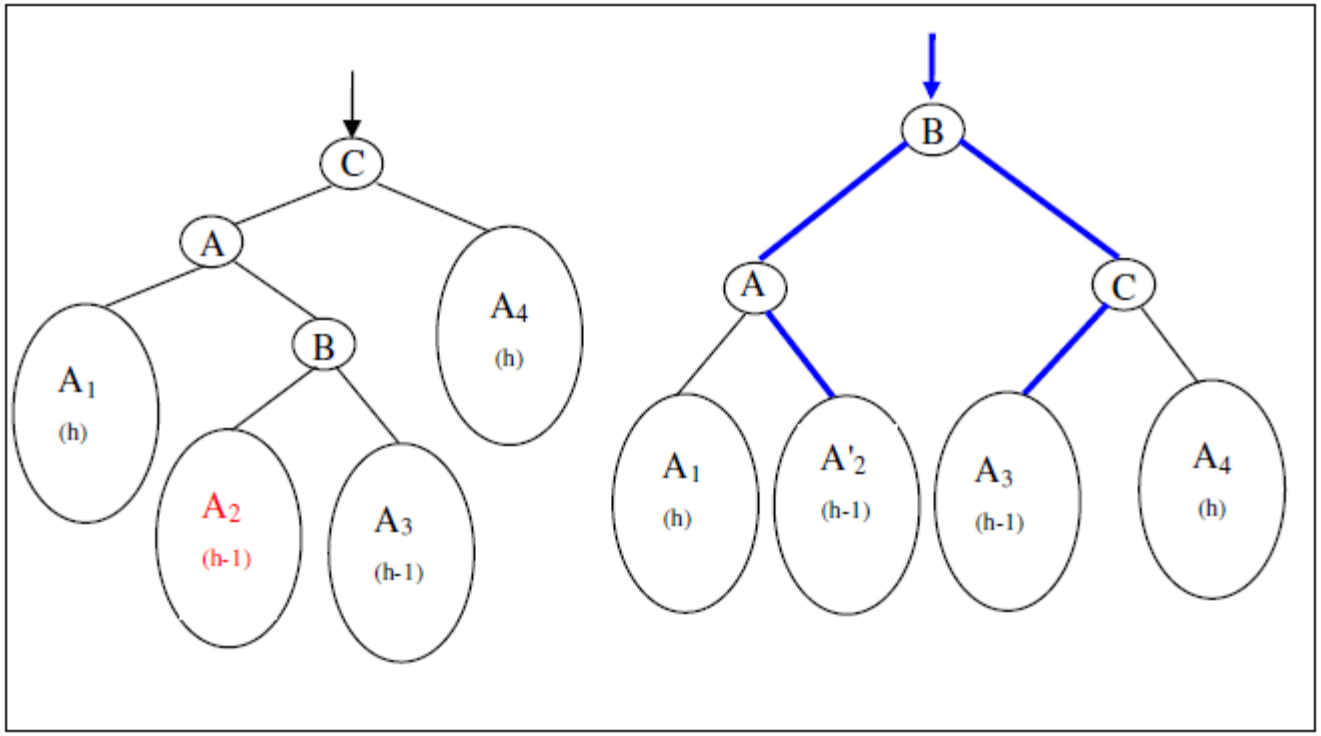
Cautarea in colectii de date ordonate

Varianta 4:



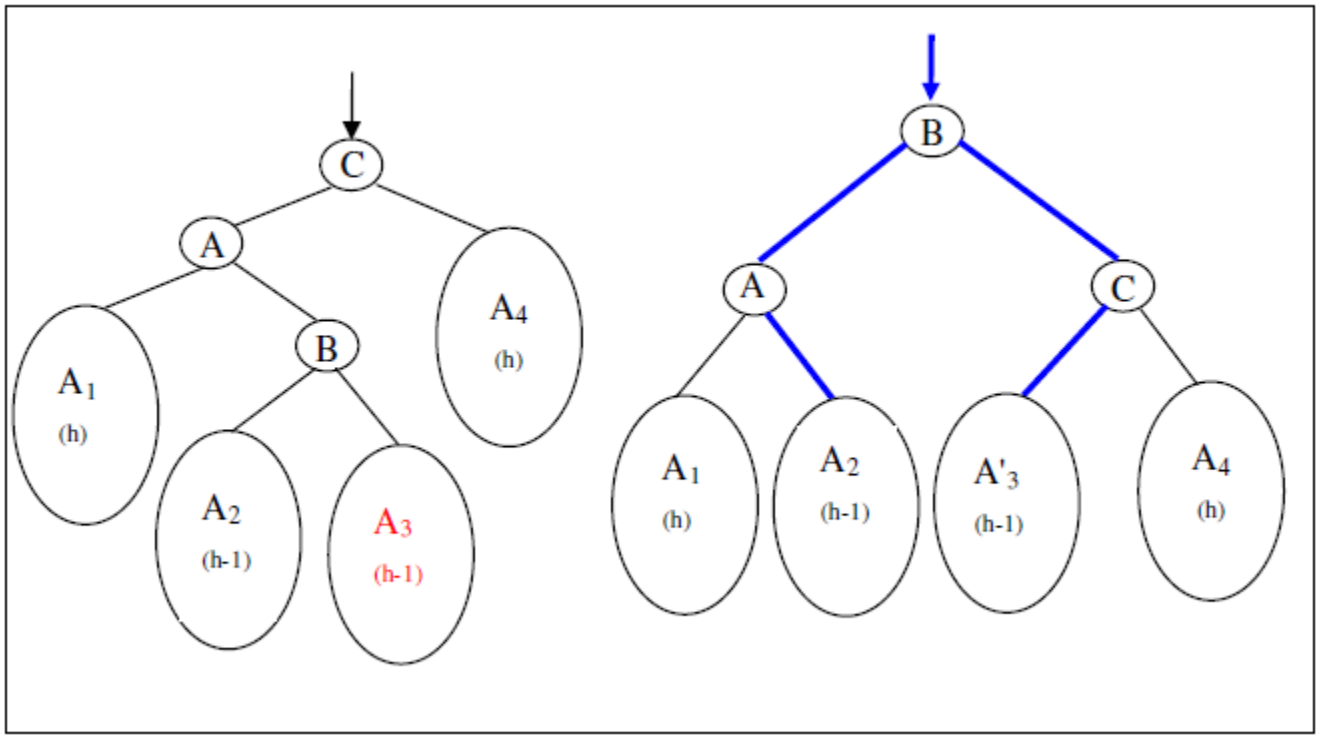
Cautarea in colectii de date ordonate

Varianta 5:



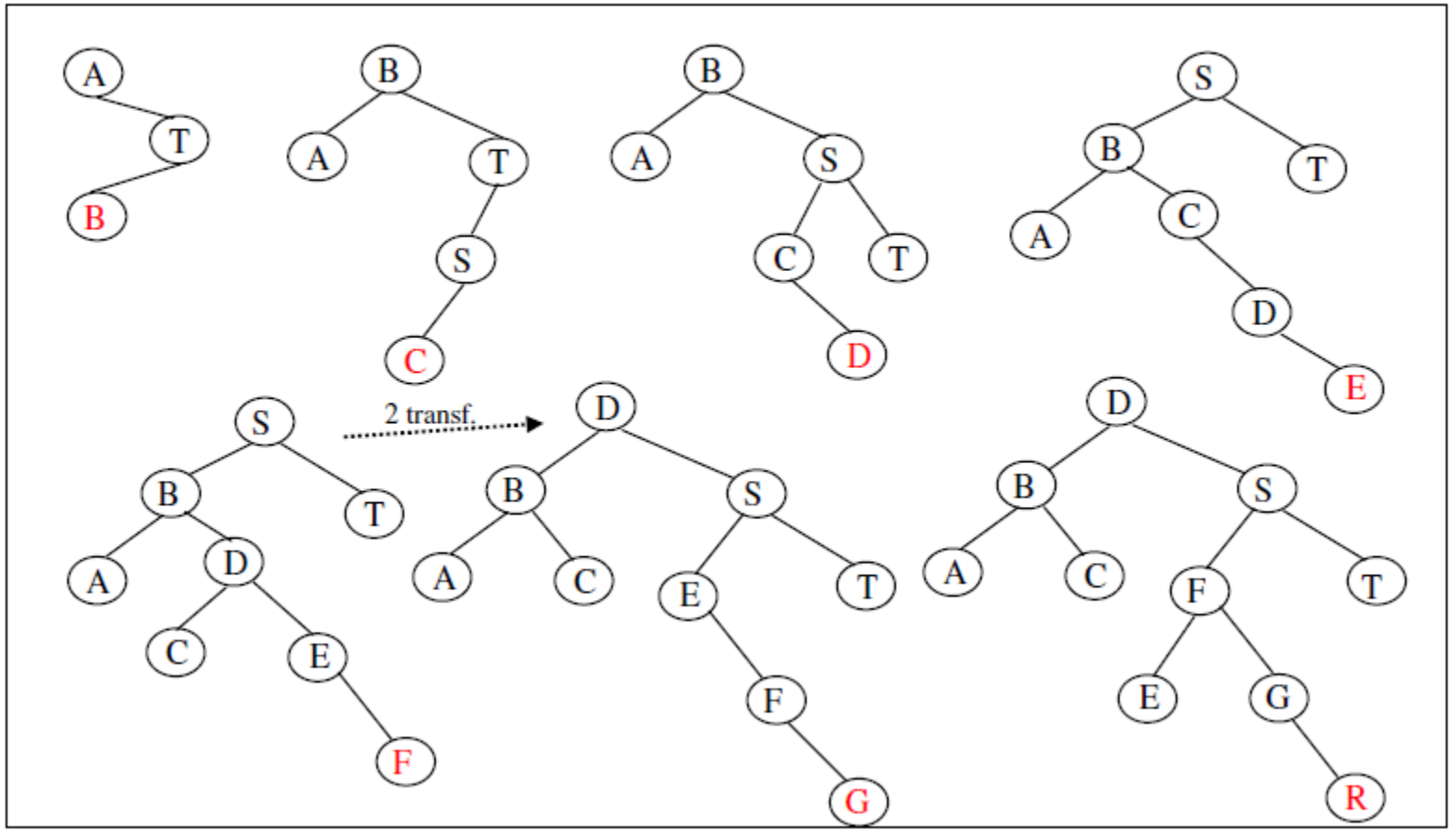
Cautarea in colectii de date ordonate

Varianta 6:

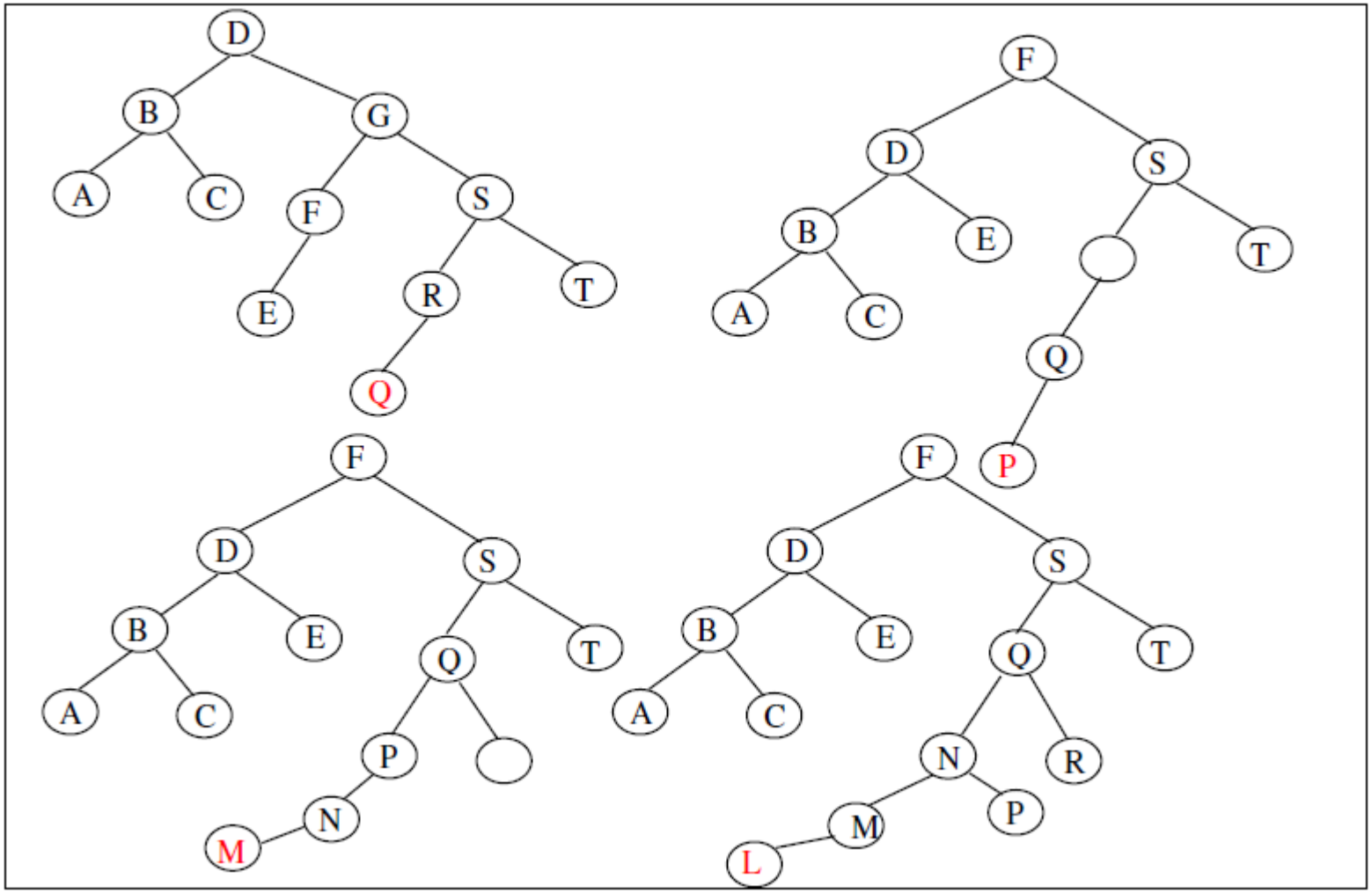


Cautarea in colectii de date ordonate

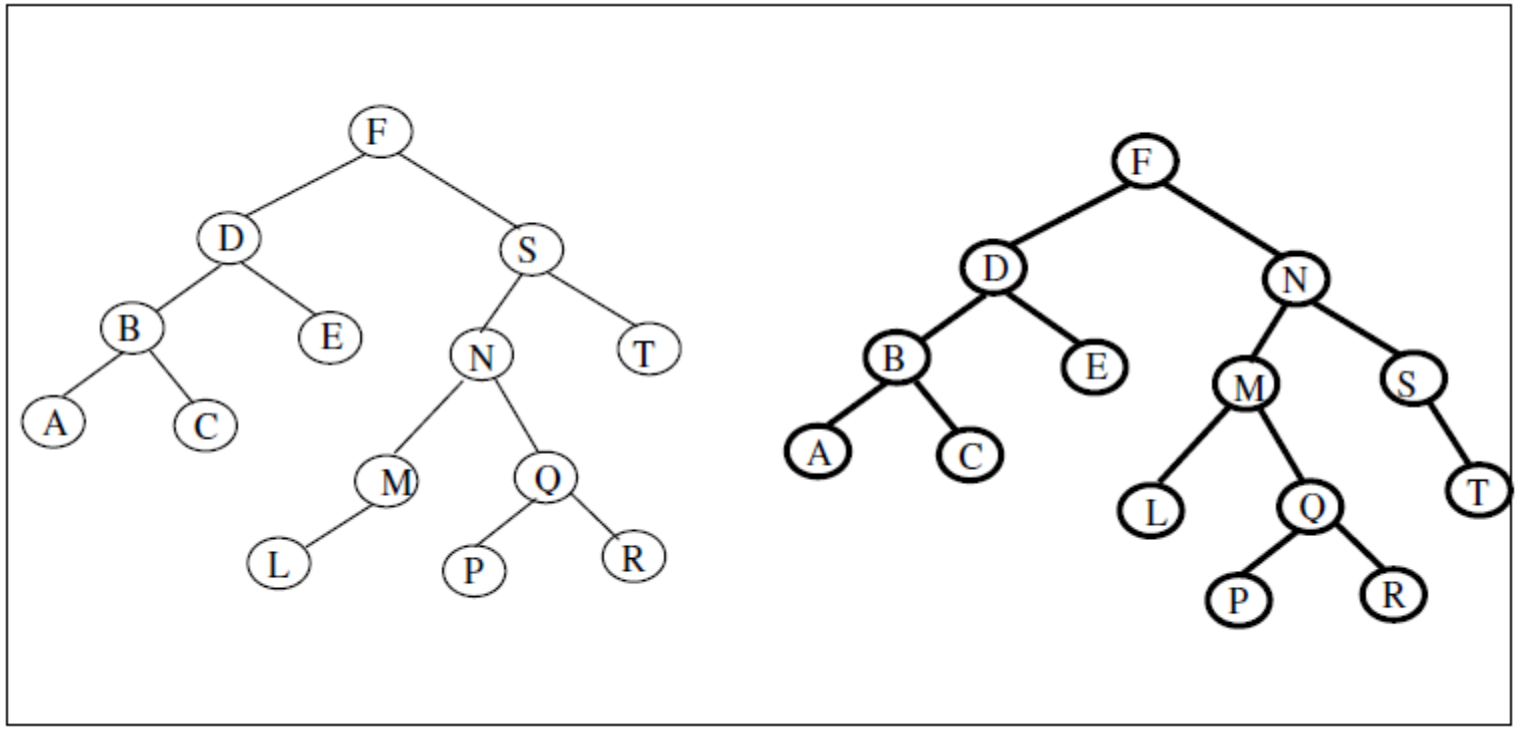
In continuare se vor arăta modurile de echilibrare pentru ultima colecţie folosită:
{A, T, B, S, C, D, E, F, G, R, Q, P, N, M, L}



Cautarea in colectii de date ordonate



Cautarea in colectii de date ordonate



Bibliografie

- [Si10] SILBERSCHATZ A., KORTZ H., SUDARSHAN S., *Database System Concepts*, McGraw-Hill, 2010, cap. 10, 11
- [El10] ELMASRI R., NAVATHE S.B., *Fundamentals of Database Systems*, 6th Ed., Pearson, 2010.
- [Ra07] RAMAKRISHNAN, R., Database Management Systems. McGraw-Hill, 2007, *cap. 7-10*
- [Kn76] KNUTH, D.E., *Tratat de programare a calculatoarelor. Sortare si cautare*. Ed.Tehnica, Bucuresti 1976
- [Ga08] GARCIA-MOLINA, H., ULLMAN, J., WIDOM, J., *Database Systems: The Complete Book,* Pearson Prentice Hall, 2008], *cap. 12-14*.
- Leon Tambulea, UBB, curs de baze de date