CĂUTAREA DUPĂ CHEI FLEXIBILE ÎN APLICAȚII DISTRIBUITE

Ion Ivan
ionivan@ase.ro

Eugen Dumitrașcu
eugen.dumitrascu@cs.ucv.ro

Daniel Milodin
daniel.milodin@ase.ro

Academia de Studii Economice București

Abstract: Se prezintă aspecte de bază privind aplicațiile distribuite: definire, particularități și importanță. Se detaliază aspecte legate de procese de căutare și regăsire în baze de date folosind tipuri de chei. Se dau mecanisme de construire a cheilor flexibile. Se prezintă proiectarea unei interfețe cu utilizatorul cu chei flexibile. Sunt realizate module de căutare după chei flexibile. Se stabilesc pașii în utilizarea cheilor flexibile în aplicații pentru magazine virtuale.

Cuvinte cheie: baze relaționale, chei flexibile, aplicații distribuite.

Abstract: There are presented base aspects regarding distributed applications: defining, particularities and importance. They are detailed aspects regarding processes of searching and finding information using different types of keys. There are given mechanisms of constructing flexible keys. There is presented the process of projecting interfaces with user based on flexible keys. There are created modules of searching based on flexible keys. There are defined steps in using flexible keys for applications regarding virtual stores.

Key words: relational databases, flexible keys, distributed applications.

1. Aplicații distribuite

O aplicație distribuită sau o aplicație globală este o aplicație ce implică accesul la date din mai multe noduri ale unei rețele de calculatoare. Componentele se execută pe noduri diferite, pe platforme diferite, conectate în rețea.

Aplicațiile distribuite sunt acelea în care mai mulți beneficiari sau utilizatori dispuși în diferite puncte în teritoriu, accesează resurse definite pentru o rețea de calculatoare în vederea soluționării unei probleme. Concepțiile moderne privind tranzacțiile bancare, tranzacțiile interbancare, efectuarea activităților de comerț electronic, activități de instruire, de informare, de testare a cunoștințelor, de încheiere de contracte on-line sunt numai câteva dintre aplicațiile distribuite, care trebuie să caracterizeze societatea informațională. Filosofiile de dezvoltare pentru e-learning, e-government, e-business, pentru organizații virtuale și implementarea noilor forme de muncă se bazează pe principiile aplicațiilor distribuite [3]. Fiecare utilizator are setul lui de informații la care are acces. Există informații la care au acces toți utilizatorii și informații la care au acces numai anumiți utilizatori. Accesul se face prin autentificarea utilizatorului cu ajutorul unui nume de utilizator și o parolă.

Particularitățile aplicațiilor distribuite sunt:

- interfețe puternice, care permit utilizarea lor de către un număr foarte diferit de cetățeni;
- grad de generalitate ridicat, care permite ca un număr foarte mare de persoane să soluționeze problemele proprii;
- interfețe prietenoase, care să permită eliminarea erorilor de introducere a datelor și abandonarea utilizării;
- niveluri de securitate, care să garanteze că sistemul de tranzacții este operațional;
- niveluri de acces, care să soluționeze convenabil problema securității cu cea a transparenței;
- nivel ridicat de corectitudine și de fiabilitate;
- asigurarea înregistrării de informații suficiente, care să dea posibilitatea reconstituirii traseelor informatice;
- componentele oricărei aplicații distribuite conțin două părți importante: partea de aplicație și partea de comunicație; unele componente conțin și o parte specială, denumită partea administrativă, cu rol de control și monitorizare a componentelor;
- grad mare de modularitate și posibilitatea de extensibilitate prin adăugarea sau eliminarea unor componente software sau hardware;
- utilizarea în comun a resurselor de către mai mulți utilizatori;
- disponibilitate mare în cazul defectării unor componente;
- siguranța în funcționare.

Termenul de aplicație distribuită are trei aspecte:

- aplicația A, a cărei funcționalitate este împărțită în *n* componente, A_1, A_2, ..., A_n, $n \in N$, $n>1$ ce interacționează și cooperează între ele; fiecare componentă fiind o aplicație sau un proces;
- componentele A_i sunt entități autonome care rulează pe calculatoare diferite;
- componentele A_i schimbă informații prin intermediul rețelei.

Importanța aplicațiilor distribuite este din ce în ce mai mare în societatea informațională. Astăzi, aproape orice aplicație se realizează distribuit.

2. Procese de căutare și regăsire în baze de date

O bază de date se definește ca fiind modul de stocare a informațiilor sau datelor pe un suport extern, cu facilitatea de regăsire a acestora. O bază de date este memorată într-unul sau mai multe fișiere. Cele mai cunoscute baze de date sunt manipulate cu ajutorul sistemelor de gestiune a bazelor de date – SGBD. Modelul cel mai răspândit de baze de date este cel relațional, în care datele sunt memorate în tabele, dar există și alte modele de memorare a bazelor de date, precum modelul ierarhic, modelul orientat-obiect și modelul XML [11].

Majoritatea aplicațiilor distribuite au un nivel în structura lor, care conține o bază de date. Baza de date este locală sau distribuită, adică este controlată de un sistem de gestiune a bazelor de date, în care dispozitivele de stocare nu sunt atașate în totalitate de o unitate centrală de prelucrare obișnuită. Ele sunt stocate pe mai multe calculatoare interconectate, așezate în aceeași locație fizică sau dispersate într-o rețea. Distribuția și tranzacțiile în bazele de date distribuite trebuie să fie transparente față de utilizator.

Cheia determină în mod unic un element al unei entități dintr-o bază de date. Aceasta este formată dintr-un atribut sau mai multe atribute. Cu ajutorul cheii, se regăsește o informație din baza de date.

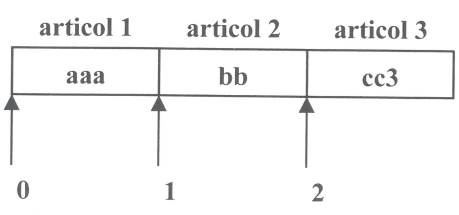
Proiectarea bazelor de date este făcută astfel încât cheia este utilizată pentru a regăsi diferite date. Un exemplu elocvent de cheie este codul numeric personal – CNP. Acesta este cheie în multe aplicații cu baze de date, în care identificarea persoanei se face după acest câmp unic, cunoscut de persoana căruia i-a fost atribuit. De aceea, codul numeric personal trebuie să stea la baza definirii cheii de căutare pentru orice aplicație distribuită și, împreună cu un șir de caractere (parola definită de utilizator), contribuie la realizarea unui proces de accesare a informației care privește pe utilizator.

În diverse aplicații având câmpul cod numeric personal se găsesc informațiile legate de acea persoană, precum: date personale, situația familiară, locul de muncă, proprietățile persoanei. Prin urmare, cu aceeași cheie se regăsesc date din diverse aplicații.

În cazul bazelor de date nerelaționale, este greu de definit o cheie unică. Astfel, există chei care permit extragerea de submulțimi, precum toate persoanele care au mașină roșie, toate persoanele care stau într-un anumit bloc. Persoanele care au același nume și prenume sunt foarte multe, dar cartea de telefon conține și adrese ce permit individualizarea. Pentru astfel de probleme, trebuie gândite procese de regăsire, care să conducă la o listă cât mai restrânsă. Însă, aceste procese trebuie să nu dureze foarte mult și să nu îngreuneze procesele care se realizează asupra bazei de date.

Se enumeră câteva metode de regăsire a informației pentru baze de date nerelaționale:

- cheile se pun în listă împreună cu poziția articolelor;



cheie	poziție
aaa	0
bb	1
cc3	2

Figura 1. Așezarea în listă a cheilor

- cheile se pun în structură arborescentă împreună cu poziția articolelor;

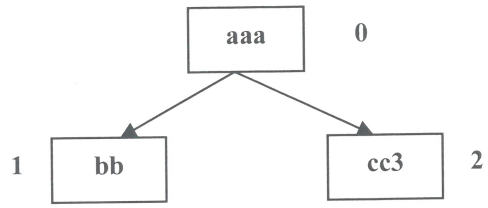


Figura 2. Aşezarea în arbore a cheilor

Sistemele de gestiune a bazelor de date au implementate unele facilităţi privind căutări foarte rapide. Cu toate acestea, apare problema căutării flexibile din cauza faptului că procesele de căutare trebuie extinse şi la alte reprezentări decât cele din cadrul fişierelor ASCII.

Conceptul de căutare în structuri de date constă în a determina numărul mediu de căutări.

Fie n seturi de date de acelaşi fel, fiecare având câte o cheie notată k_i cu $i \in [1, n]$, ca în figura 3.

	k_1	k_2	k_3
indice	**1**	**2**	**3**
poziţie	**1**	**2**	**3**

Figura 3. Set de date

Numărul de citiri şi comparări pentru a ajunge la k_j este j.

Numărul de căutări pentru cheia k_j este f_j.

Pentru căutarea uniformă, rezultă relaţia: $f_1 = f_2 = \ldots = f_n = F$.

Numărul total de căutări este dat de relaţia: $NTC_u = F \cdot (1 + 2 + \ldots + n) = F \cdot \frac{n \cdot (n+1)}{2}$.

Numărul mediu de căutări se calculează folosind formula: $NMC_u = \frac{F}{n} = F \cdot \frac{n+1}{2}$.

Pentru o căutare oarecare $f_1 \neq f_2 \neq \ldots f_n \neq f_1$. Deci, numărul total de căutări este: $NTC = \sum_{i=1}^{n} i \cdot f_i$.

Numărul mediu de căutări se calculează cu formula: $NMC = \frac{\sum_{i=1}^{n} i \cdot f_i}{n}$

Dacă se sortează descrescător după numărul de căutări pentru fiecare cheie atunci: $f_{d1} > f_{d2} > \ldots > f_{dn}$ număr comparaţii 1.....2.....n unde f_{dj} este numărul de căutări pentru cheia k_j.

Se calculează numărul total de căutări în cazul descrescător:

$$NTC_d = \sum_{i=1}^{n} i \cdot f_{di}$$

unde $f_{d1} > f_{d2} > \ldots > f_{dn}$

Se calculează numărul mediu de căutări cazul descrescător:

$$NMC_d = \frac{\sum_{i=1}^{n} i \cdot f_{di}}{n}$$

unde $f_{d1} > f_{d2} > \ldots > f_{dn}$

Se demonstrează prin exemplificări că $NTC_d \leq NTC$ şi $NMC_d \leq NMC$.

Se consideră situația:

Chei	10	20	30	40
Nr. căutări	10	5	20	100
Nr. comparații	1	2	3	4

$$NTC = 1 \cdot 10 + 2 \cdot 5 + 3 \cdot 20 + 4 \cdot 100 = 480$$

$$NMC = \frac{480}{4} = 120$$

După sortarea descrescătoare a numărului de căutări, se obține:

Chei	40	30	10	20
Nr. căutări	100	20	10	5
Nr. comparații	1	2	3	4

$NTC_d = 1 \cdot 100 + 2 \cdot 20 + 3 \cdot 10 + 4 \cdot 5 = 190$.

$NMC_d = \frac{190}{4} = 47{,}5$.

Deci, $NTC_d < NTC$ şi $NMC_d < NMC$.

Fie n seturi de date de acelaşi fel. aşezate în structură arborescentă, fiecare având câte o cheie notată k_i cu $i \in [1, n]$, figura 4.

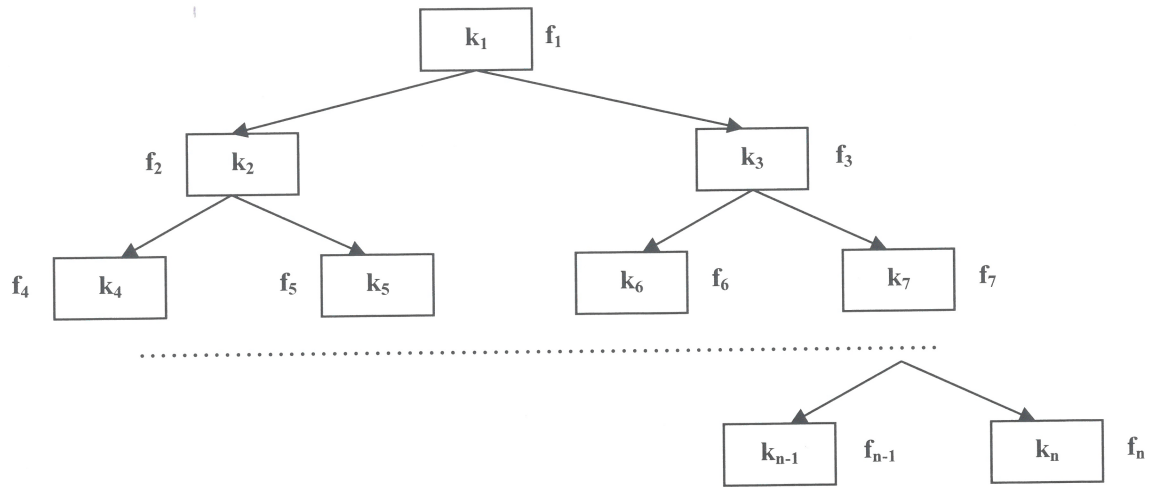


Figura 4. Set de date în structură arborescentă

Numărul de citiri şi comparări pentru a ajunge la k_j este niv, unde niv este nivelul pe care se află cheia. Numărul total de niveluri este Nrniv.

$n = 2^{Nrniv} - 1 \Rightarrow Nrniv = \log_2 (n+1)$

Numărul de căutări pentru cheia k_j este f_j.

Numărul total de căutări este:

$$NTC_a = 1 \cdot f_1 + 2 \cdot (f_2 + f_3) + ... + Nrniv \cdot (f_{2^{Nrniv-1}} + ...f_{2^{Nrniv}-1}) = \sum_{i=1}^{Nrniv} i \cdot \sum_{j=2^{1-1}}^{2^i-1} f_j = \sum_{i=1}^{\log_2(n+1)} i \cdot \sum_{j=2^{1-1}}^{2^i-1} f_j .$$

Numărul mediu de căutări se calculează cu formula:

$$NMC_a = \frac{NTC_a}{n} = \frac{\sum_{i=1}^{\log_2(n+1)} i \cdot \sum_{j=2^{1-1}}^{2^i-1} f_j}{n}$$

În organizarea arborescentă a cheilor, numărul total de căutări şi numărul mediu de căutări sunt diferite față de cele în organizarea listă.

În vederea obținerii de valori minime pentru numărul total de căutări și numărul mediu de căutări în structura arborescentă, se vor așeza cheile într-un arbore binar astfel încât, pornind de la rădăcină către frunză, valorile numărului de căutări pentru fiecare cheie să fie în ordine descrescătoare pe fiecare nivel.

Se consideră situația exemplificată mai sus și pentru aceasta arborele atașat este cel din figura 5:

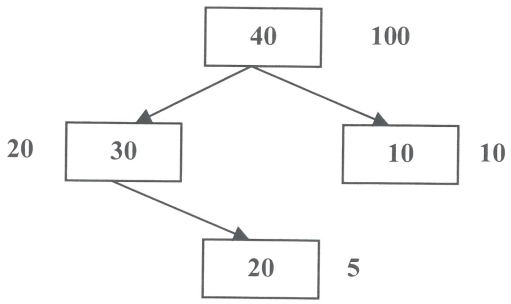


Figura 5. Așezarea în arbore a cheilor

$NTC_a = 1 \cdot 100 + 2 \cdot 20 + 2 \cdot 10 + 3 \cdot 5 = 175$.

$NMC_a = \frac{175}{4} = 43.75$.

Deci, $NTC_a \leq NTC_d \leq NTC$ și $NMC_a \leq NMC_d \leq NMC$.

3. Construirea de chei flexibile

Construirea cheilor flexibile presupune crearea unor chei a căror structură se modifică pentru a găsi informațiile dorite într-o bază de date nerelațională.

Șir cuvinte		Procedeu	Cheie rezultată
Cuvânt1	**Cuvânt2**		
Popescu	Ion	preluare	Popescu Ion
Popescu	Ion	inversare	Ion Popescu
Popescu	Ion	concatenare	PopescuIon
Popescu	Ion	inversare și concatenare	IonPopescu
Popescu	Ion	concatenare și normalizare	popescuion
Popescu	Ion	inversare, concatenare și normalizare	ionpopescu
Popescu	Ion	asociere echivalențe	Popescu Nelu Popescu Jan Popescu Jean Popescu Ionuț Popescu Ionel

Cheile flexibile au următoarele proprietăți:

a) cheia este un șir de cuvinte

	Nume	**Prenume**
Cheia	Popescu	Ion

b) cuvintele sunt comutative

	Prenume	**Nume**
Cheia	Ion	Popescu

c) cuvintele se concatenează

	NumePenume
Cheia	PopescuIon

	PenumeNume
Cheia	IonPopescu

d) unele cuvinte formează o familie:

Prenume derivate: Gheorghe, Gică, Gigi, Gicu, George, Georgel, Georgică.

e) din şir se construiesc subşiruri:

Şirul: Munteanu

Subşiruri: Muntean, Munte

f) un cuvânt este rădăcină şi apar derivate:

Cuvânt rădăcină: Dum

Cuvinte derivate prin adăugări de sufixe: Duma, Dumitru, Dumitrescu, Dumitra, Dumitrana

Mecanismele de creare a cheilor flexibile folosesc proprietăţile acestora menţionate mai sus:

- dacă cheia este formată dintr-un singur cuvânt, se folosesc proprietăţile d), e) şi f);
- dacă cheia este formată din mai multe cuvinte, se folosesc proprietăţile a), b), c).

O metodă de construire de cuvinte prin adăugări de sufixe este metoda cu structuri de date de tip arbore TRIE. Arborele TRIE este un arbore asociat unui automat finit care recunoaşte o secvenţă.

În figura 6, se prezintă arborele asociat cuvintelor: Duma, Dumitra, Dumitrana, Dumitrescu.

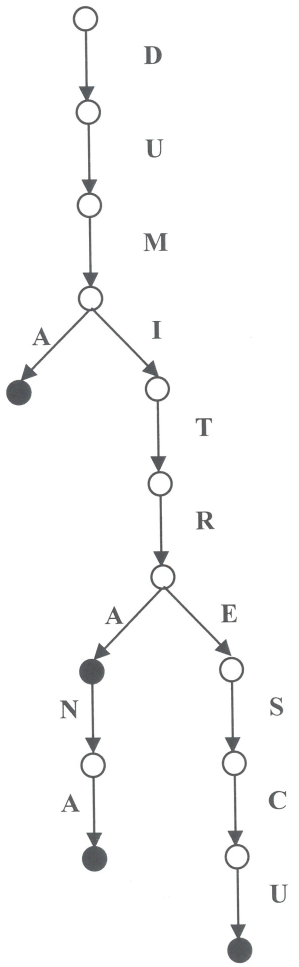


Figura 6. Arborele TRIE

Acest model de creare a unei chei flexibile este util de aplicat în aplicaţii distribuite cu baze de date, care conţin informaţii legate de persoane, precum cartea de telefon on-line.

g) proiectarea interfeţei pentru lucru cu chei flexibile:
Pentru proiectarea unei interfeţe cu utilizatorul cu ajutorul cheilor flexibile trebuie să se ţină seama de următoarele:

- formularul trebuie să conțină câmpuri obligatorii şi câmpuri opționale;
- câmpurile obligatorii formează şiruri de cuvinte de căutare;
- cuvintele obligatorii trebuie validate, data trebuie să aibă formatul *ZZ-LL-AAAA*, unde LL $\in$ [1, 12], ZZ $\in$ [1, X_{LL}], $X_{LL} \in \{28, 29, 30, 31\}$, AAAA $\in$ [1870, 2007];
- se trimit datele şi mesajul primit vizează scrierea cu altă culoare a câmpurilor din formular incorecte şi eventual mesaje de ce sunt incorecte;
- existența unui buton de efectuare a corecțiilor şi unul de anulare a datelor culese;
- interfața trebuie să includă multe liste din care utilizatorul alege, cum sunt nume de străzi, nume de orașe, nume de instituții, valori;
- ordinea de introducere a datelor;
- exemple de completare a câmpurilor;
- se caută să se excludă caracterul repetitiv al introducerii aceluiaşi câmp;
- definirea de chei pentru autentificare (*username* şi *parola*); flexibilitatea apare din faptul că se dă utilizatorului posibilitatea de a schimba periodic *username* şi *parola*.

Se prezintă o variantă de astfel de formular de introducerea a datelor pentru înregistrarea la simpozion cu lucrări.

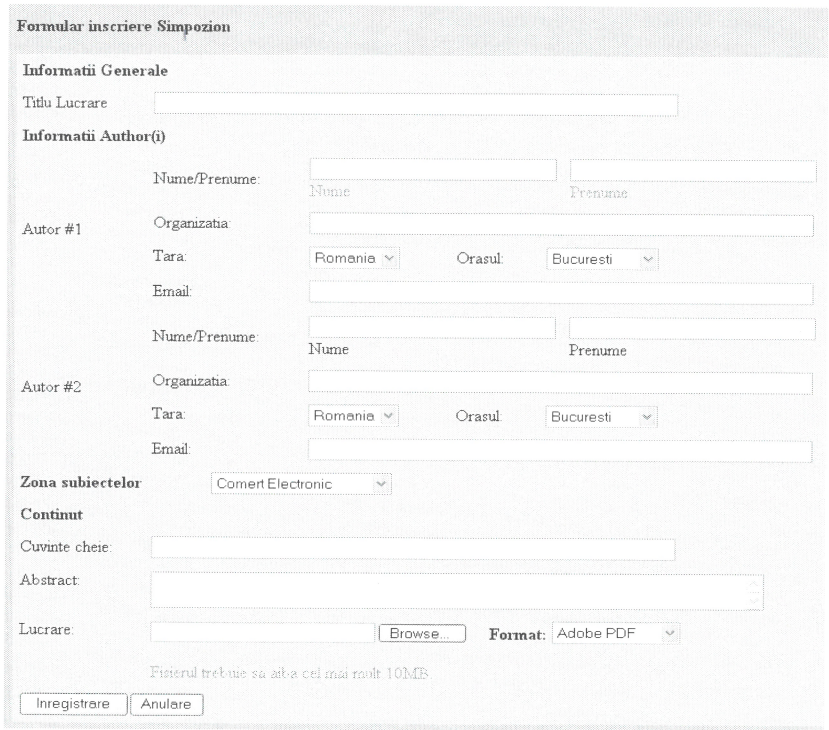


Figura 7. Formular de înscriere simpozion

Respectarea condițiilor de proiectare a interfeței asigură o validare superioară a datelor introduse şi creşterea eficienței de preluare a datelor în cadrul aplicației.

h) module pentru încărcarea bazelor de date destinate accesului cu chei flexibile:

Există diverse modele pentru popularea bazelor de date destinate accesului cu chei flexibile.

În bazele de date se stochează datele despre elementele unei colectivități, precum şi indicii care ajută la regăsirea informațiilor, cum sunt cheile şi pozițiile în baza de date. Se sortează cheile şi se păstrează un istoric al operațiilor efectuate. La fiecare modificare a datelor, prin introducere sau actualizare, se reia sortarea cheilor. Dacă datele nu s-au modificat şi se încearcă sortarea, aceasta nu se mai face deoarece cheile sunt deja sortate. Această metodă prin care se memorează istoricul operațiilor, cum este operația de sortare, se aplică des în cazul bazelor de date cu acces prin chei flexibile.

Se consideră o tabela cu chei şi poziții:

Chei	17	3	20	1	2
poziție	0	1	2	3	4

Sortarea descrescătoare a tabelei după chei înseamnă:

Chei	20	17	3	2	1
poziție	2	0	1	4	3

La adăugarea unei chei noi aceasta în tabela inițială aceasta devine:

Chei	17	3	20	1	2	15
poziție	0	1	2	3	4	5

După sortare se obține următoarea dispunere:

Chei	20	17	15	3	2	1
poziție	2	0	5	1	4	3

În felul acesta, căutarea începe cu prima cheie din tabela sortată descrescător, iar în cazul în care informația nu este găsită, se apelează la următoarea cheie sau la o combinație a celor două chei, și procedeul continuă până la găsirea informației dorite.

i) module pentru căutarea după chei flexibile:

Pentru căutarea informațiilor după chei flexibile, se construiesc module care vor urma algoritmi de filtrare a datelor prin concatenarea la cheia inițială de noi cuvinte.

Se definește operația de concatenare notată _.

Fie 2 cuvinte cuv1 și cuv2 atunci cuv1_cuv2=cuv1cuv2.

Se definește operația de interschimbare notată $\leftrightarrow$.

Fie 2 cuvinte cuv1 și cuv2 atunci cuv1$\leftrightarrow$cuv2=>cuv1=cuv2 și cuv2=cuv1.

Se construiesc vectori V_k de articole cu elemente perechi de forma (cuv, poz), unde cuv – cuvântul cu proprietate de cheie flexibilă și poz – poziția articolului în baza de date.

V_k={ (cuv, poz) } cu k > 0

Acești vectori se memorează în fișiere. La adăugare de articol în baza de date, se operează pe acești vectori cu inserare de elemente. La începerea lucrului, vectorii stau la baza construirii arborelui de căutare, ce permite găsiri rapide ale poziției articolului.

V_1 este foarte rigid 1:1 la o cheie găsim un articol.

V_k, cu k >1, este cel mai flexibil 1:x:y la o cheie inițială se generează x chei de căutare cu care se găsesc y articole, se solicită informații suplimentare care filtrează din aproape în aproape cele y articole până se ajunge la unul singur.

Se consideră că în baza de date există câmpurile: nume, prenume, adresa, data nașterii.

Se construiesc vectorii:

V_1 = {(nume_prenume, poz) }

V_2 = { (prenume_nume, poz) }

V_3 = { (nume_prenume_adresa, poz) }

V_4 = { (prenume_nume_adresa, poz) }

Algoritmul este prezentat în figura 8, pașii parcurși fiind următorii:

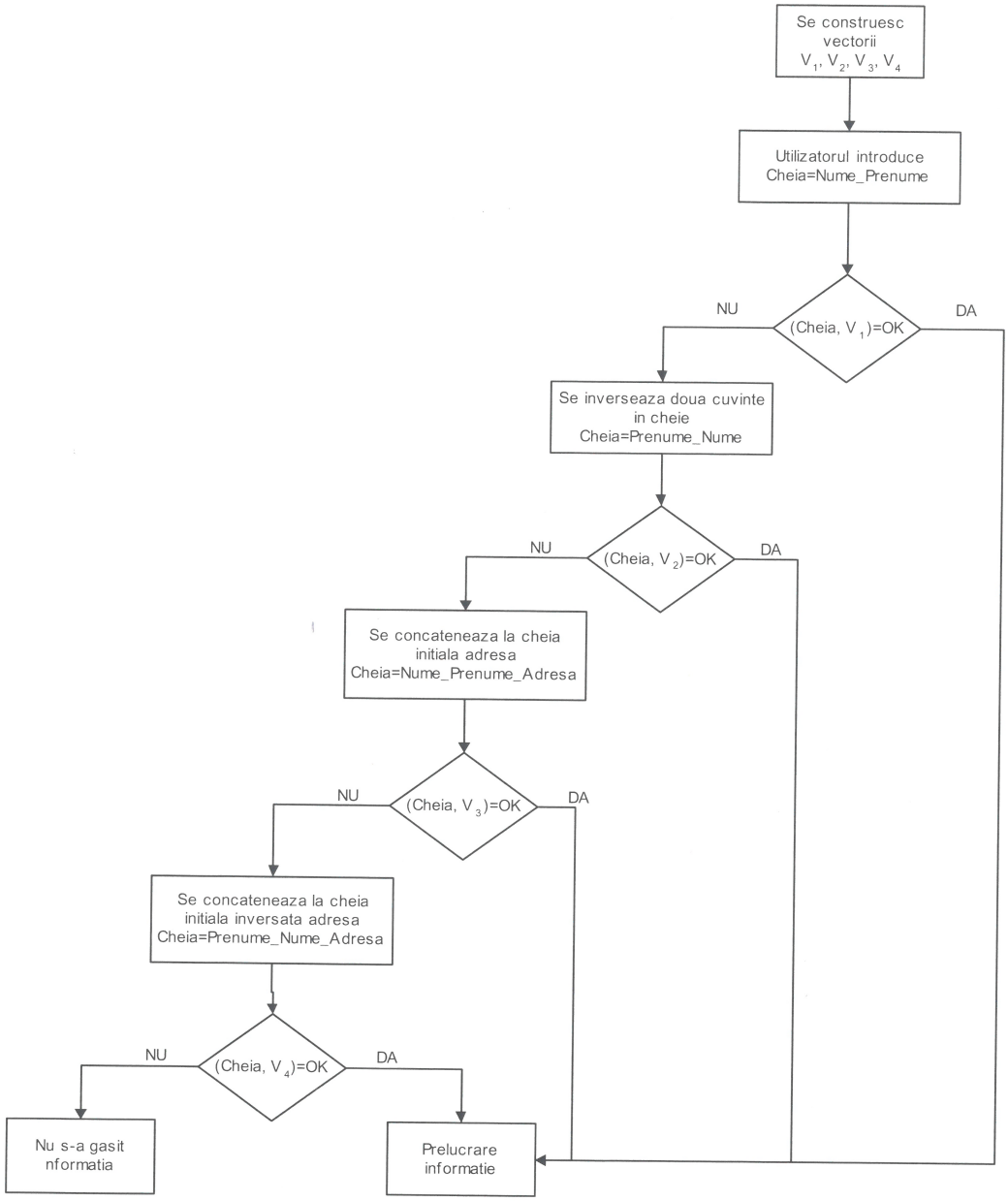


Figura 8. Secvență de căutare cu cheie flexibilă

- cheie inițială: PopescuGheorghe => 200 de persoane cu acest nume;
- se cere strada pe care locuiesc: BulevardulRepublicii => Cheia = PopescuGheorgheBulevardulRepublicii => 5 persoane;
- se cere data naşterii, în format ZZ-LL-AAAA: 12-12-1970 => Cheia = PopescuGheorgheBulevardulRepublicii12-12-1970 => 2 persoane;
- se cere numărul de la apartament=12 => Cheia = PopescuGheorgheBulevardulRepublicii12-12-197012 => 1 persoană.

S-a definit un proces de filtrare prin concatenarea la o cheie de noi cuvinte.

Algoritmul de filtrare prin concatenare se generalizează în felul următor:

P1: se consideră o cheie S_1 formată din h cuvinte care găseşte N_1 înregistrări în baza de date.

P2: se consideră o cheie S_2 formată din h+1 cuvinte care găseşte N_2 înregistrări în baza de date, unde $S_1 \subset S_2$ şi $N_1 \geq N_2$.

P3: se consideră o cheie S_3 formată din h+2 cuvinte care găseşte N_3 înregistrări în baza de date, unde $S_2 \subset S_3$ şi $N_2 \geq N_1$.

Pq: se consideră o cheie S_q formată din h+(q-1) cuvinte care găseşte N_q înregistrări în baza de date, unde $S_{q-1} \subset S_q$ şi $N_{q-1} \geq N_q$, cu $N_q \in \{0,1\}$.

La ultimul pas al algoritmului, N_q este 1 atunci când s-a găsit o singură înregistrare şi anume, cea căutată, sau N_q este 0 atunci când nu s-a găsit nici o înregistrare în baza de date.

În concluzie, $S_1 \subset S_2 \subset S_2 \ldots \subset S_q$ şi $N_1 \geq N_2 \geq N_3 \ldots \geq N_q$, cu $N_q \in \{0,1\}$.

j) utilizarea cheilor flexibile în dezvoltarea aplicaţiilor de magazine virtuale:

Utilizarea cheilor flexibile are o largă aplicabilitate în cadrul aplicaţiilor distribuite web de comerţ electronic. În cazul aplicaţiilor care conţin diferite baze de date cu produse, căutarea unui produs se face printr-un proces de concatenare prin cheie flexibilă.

Magazinele virtuale reprezintă un tip de aplicaţie de comerţ electronic pentru care acest proces de căutare se aplică cu succes.

Algoritmul de formare a cheii de căutare pentru o astfel de aplicaţie este următorul:

- se consideră clasele de produse $P_1, P_2 \ldots, P_k$;
- se selectează clasa P_i şi apare o listă de tipuri de produse $TP_{i1}, TP_{i2}, TP_{i3}, \ldots TP_{ij}, \ldots TP_{in}$;
- se selectează tipul de produse TP_{ij}, apare o listă de mărci de produse;
- se selectează marca şi se face comanda.

La fiecare pas al selecţiei la cheia selectată iniţial, se concatenează pe rând câte un alt cuvânt care determină în final cheia de regăsirea a produsului căutat prin procedeul de obţinere a cheii primare.

Conform cu algoritmul de căutare cu cheie flexibilă, produsele P_i cu $i \in [1, k]$ formează vectorul V_1, produsul ales concatenat cu tipurile de produse P_i _TP_{ij}, cu $j \in [1, n]$ formează un vector V_2, iar aceasta cheie concatenată cu marca formează V_3.

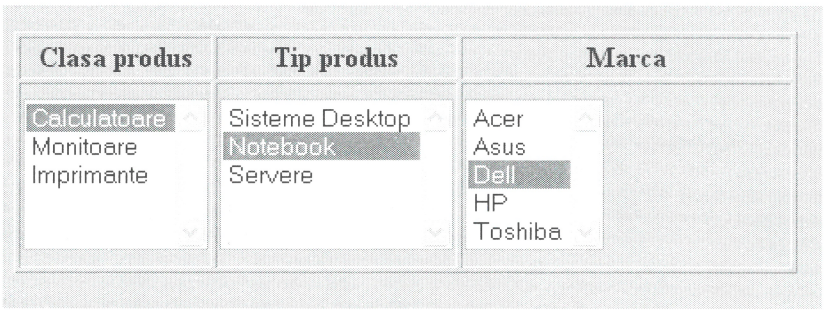


Figura 9. Interfaţă cu căutare cu cheie flexibilă

În cazul în care se adoptă o altă strategie:

- utilizatorul tastează o serie de caractere conţinute de categoria ce urmează a fi selectată;
- apare o listă cu cuvintele ce conţin acele caractere;
- pe măsură ce mai adaugă caractere lista se scurtează;
- dacă utilizatorul vede în listă ceea ce îl interesează, selectează;
- căutarea se face după cheia construită în acest fel, care este cheie unică; procesul de construire este flexibil şi găseşte cheia prin adăugarea unor sufixe la cheia iniţială, formată din primul caracter tastat.

Lista derulantă găseşte informaţia printr-un algoritm ce utilizează un arbore TRIE.

O altă aplicabilitate a procesului de căutare prin chei flexibile îl are căutarea unor persoane în baza de date a populaţiei.

Se caută o persoană numită Popescu Gheorghe, dar utilizatorul introduce Gică Popescu. Dacă se găseşte, procesul de opreşte, dacă nu, se caută Popescu Gică şi, la fel, dacă se găseşte, procesul de căutare se opreşte, dacă nu, se continuă.

Se creează o listă ordonată descrescător, după ordinea frecvenţelor prenumelor derivate din cheia de căutare sau a prenumelor similare.

Prenume	Gheorghe	George	Gigi	Gicu	Georgel	Georgică	Gogu	Goguţă
Frecvenţă	1000	700	100	60	30	20	10	3

Se construiesc rând pe rând chei de căutare, formate din nume şi variantele de prenume din listă. Mai întâi *Popescu Gheorghe* pentru că *Gheorghe* are frecvenţa 1000 dacă se găseşte s-a încheiat căutarea, dacă nu, se construieşte *Gheorghe Popescu*, şi tot aşa, se iau în ordine descrescătoare după frecvenţă toate prenumele.

Când se face căutarea, se contorizează tipurile de situaţii întâlnite:

- număr de cazuri în care s-a dat prenumele şi numele;
- inversarea nume-prenume;
- număr de cazuri în care s-a dat un derivat al prenumelui şi numele;
- număr de cazuri în care s-a dat un derivat al prenumelui şi numele şi apoi s-a făcut inversarea;
- număr căutare după nume şi prenume foarte frecvent.

În cazul în care căutările eşuează, se iau situaţiile în care a trebuit să se completeze cu nume de stradă dat corect sau incorect, se merge pe similitudine cu numele de stradă existent în baza de date.

Tot ceea ce vizează un istoric al procesului de căutare trebuie folosit pentru rafinarea căutărilor, pentru perfecţionarea bazelor de date şi a vectorilor cu chei şi poziţie articol şi, într-o situaţie mai specială, pentru dotarea aplicaţiei cu elemente de auto-perfecţionare, adică se reordonează o structură dinamică după frecvenţe. Chiar prenumele are asociat lângă el o frecvenţa şi, eventual, numele.

Acest model se aplică şi în cazul căutărilor în baze de date ale cărţilor de telefon, în care se caută numerele de telefon ale unor persoane date prin chei formate din nume şi, eventual, şi prenume.

4. Concluzii

În viitor, problema cheilor flexibile devine şi mai actuală întrucât se pune problema de a căuta informaţii în reuniuni virtuale de baze de date. Astfel, utilizatorul care caută o informaţie incompletă, face supoziţii şi oferă şiruri de cuvinte. De aceea, cheile flexibile stau la baza motoarelor de căutare.

O altă utilizare a cheilor flexibile este în regăsirea informaţiei în baze de date multimedia. Căutarea trebuie realizată pe fişiere tip .jpg, .gif, .bmp. Într-o bază de date cu imagini, la stocare se extrag din poză părţi care corespund chipurilor de persoane în situaţia în care căutarea după chei se face în mod complex.

Calitatea căutărilor este dată de:

- lungimea listei de înregistrări – cu cât lista este mai mare cu atât timpul de căutare este mai mare;
- ordinea elementelor în listă – se preferă ordonarea după anumite frecvenţe de apariţie a înregistrărilor;
- omogenitatea elementelor din listă – elementele incluse în listă trebuie să fie omogene.

Căutările cu chei flexibile analizează şi ortogonalitatea între entităţi şi gruparea acestora prin separarea lor pe baza ortogonalităţii.

Ortogonalitatea măsoară diferenţa dintre două entităţi. Dacă datele sunt complet diferite, atunci ele sunt ortogonale. Forma pe care o iau indicatorii de ortogonalitate este strâns legată de tipul şi complexitatea datelor comparate şi specificul analizei [9].

S-a observat că procesul de căutare este mai rapid, atunci când cheia flexibilă este mai complexă, şi prezintă în componenţa ei mai multe date care să identifice înregistrarea căutată. De asemenea, procesul de căutare este mai greu, atunci când cheia este mai simplă, şi duce la găsirea mai multor înregistrări şi, în acest fel, se trece la prelucrări asupra cheii, precum inversare, concatenare cu alte cuvinte, pentru a obţine o cheie complexă.

Caracteristicile importante ale algoritmilor de căutare cu chei flexibile sunt:

- precizia – depinde de diferiţi factori precum frecvenţa de căutare sau numărul de regăsiri;

- scalabilitatea – dată de timpul de accesare şi numărul de accesări a datelor;
- compararea – dată de numărul de comparări cu date deja accesate.

Articolul se adresează problemelor de regăsire a înregistrărilor în baze de date, folosind chei flexibile, care se formează pe parcursul căutărilor, prin concatenarea de noi cuvinte la cheia de căutare inițială, prin inversarea cuvintelor în cadrul cheii sau prin similaritate.

Căutarea cu chei flexibile este un mod eficient de regăsire a informației prin similaritate. Metodele de căutare cu chei flexibile vor sta la baza motoarelor de căutare sau a regăsirii informației în baze de date multimedia.

Bibliografie

1. **CYRAN, M., P. LANE**: Oracle Database Concepts, 10g Release 1, Oracle Corporation, 2003.
2. **DATE, C. J.**: An Introduction to Database Systems. Eight Edition, Pearson Addison-Wesley, 2004.
3. **IVAN, I., E. DUMITRAŞCU, M. POPA**: Evaluating the Effects of the Optimization on the Quality of Distributed Applications, Economic Computation and Economic Cybernetics Studies and Research, Vol. 40, No. 3-4, 2006, pp. 73 – 85. ISSN 0424-267X.
4. **JIANG, Z., A. JOSHI, R. KRISHNAPURAM, L. YI**: Retriever: Improving Web Search Engine Results Using Clustering, www.citeseer.com, 2000.
5. **BAI, J., JIAN-YUN NIE, G. CAO, H. BOUCHARD**: Using Query Contexts in Information Retrieval. Proc. of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2007, Amsterdam, The Netherlands, pp. 15-22.
6. **KRAFT, D. H., F. E. PETRY, W. P. BUCKLES, T. SADASIVAN**: The use of the genetic programming to build queries for information retrieval, Proc. of the 1994 IEEE Word congress on Computational Intelligence, pp. 468-473, Orlando, Florida, USA, IEEE Press.
7. **HSU, M – H., HSIN-HSI CHEN**: Information Retrieval with Commonsense Knowledge. Proc. of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2006, Seattle, Washington, USA, pp. 651 – 652.
8. **PERRONE, M. P., G. F. RUSSELL, A. ZIQ**: Machine Learning in a Multimedia Document Retrieval Framework, IBM System Journal, Vol. 41, No. 3, 2003.
9. **POPA, M.**: Evaluarea calității entităților text. Teorie şi practică, Editura ASE, Bucureşti, 2005.
10. **TELFORD, R., R. HORMAN, S. LIGHTSTONE, N. MARKOV, S. O'CONNELL, G. LOHMAN**: Usability and Design Considerations for an Autonomic Relational Database Management System, IBM System Journal , Vol. 42, No. 4, 2003.
11. * * *: http://ro.wikipedia.org/wiki/Bază_de_date.