UNIVERSITATEA DIN CRAIOVA
FACULTATEA DE AUTOMATICĂ, CALCULATOARE ŞI ELECTRONICĂ

TEZĂ DE DOCTORAT

ALGORITMI PENTRU CREŞTEREA VITEZEI DE TRANSFER AL INFORMAȚIEI ÎN BAZE DE DATE DISTRIBUITE

- REZUMAT -

Doctorand
ADRIAN RUNCEANU

Conducător ştiinţific:
Prof. Univ. Dr. Ing. MIRCEA PETRESCU

2009

Această teză prezintă câteva implementări legate de problema fragmentării datelor în general, iar în caz particular proiectarea partiționării verticale atât pentru bazele de date distribuite relaționale, cât şi pentru bazele de date distribuie orientate-obiect, şi fragmentarea orizontală împreună cu alocarea datelor în cazul bazelor de date distribuite orientate-obiect..

1. Astfel s-a studiat aplicarea câtorva algoritmi de grupare pentru procesul de proiectare a bazelor de date distribuite. Anumiți algoritmi, cum a fi algoritmul erorii-pătrate au fost adaptați şi implementați pentru fragmentarea verticală a datelor.

2. S-a prezentat o funcție obiectivă pentru n partiții, compusă din doi termeni care furnizează o evaluare bună pentru micşorarea costului proceselor de tranzacții.

3. Funcția prezentată se poate modifica prin adăugarea unor informații noi legate de tipurile de interogări (de actualizare sau de regăsire de date), de informații legate de alocarea datelor în rețea şi de costul procesului de acces la datele aflate la distanță.

4. Alocarea fragmentelor s-a prezentat printr-o abordare euristică care propune – în cazul bazelor de date distribuite orientate-obiect – o variantă care combină fragmentarea instanțelor claselor în acelaşi timp cu alocarea datelor pe diferite site-uri.

Teza de doctorat este structurată în 7 capitole urmate de anexe şi bibliografie.

În **Capitolul 1 „Introducere"** se prezintă principalele aspecte legate de proiectarea bazelor de date distribuite.

Procesul de proiectare a bazelor de date distribuite este un proces de optimizare care necesită obținerea unor soluții la câteva probleme care se întrepătrund, şi anume fragmentarea datelor, alocarea datelor şi optimizarea locală. Fiecare problemă poate fi rezolvată cu ajutorul unor abordări diferite şi din această cauză proiectarea bazelor de date distribuite devine o activitate foarte dificilă. De obicei faza de proiectare este prin definiție de tip euristic.

Proiectarea bazelor de date distribuite derivă din proiectarea bazelor de date non-distribuite doar în ceea ce priveşte aspectul distribuit. Faza de proiectare implică achiziții de date, partiționarea bazei de date, alocarea şi replicarea partițiilor şi optimizarea locală. Partiționarea bazei de date se poate efectua în câteva modalități diferite: partiționare verticală, orizontală şi hibridă (numită şi mixtă). Acest aspect al proiectării bazelor de date va fi evidențiat în această teză şi anume dezvoltarea unui algoritm de verificare a diferiților algoritmi propuşi în literatura de specialitate în cazul partiționării (fragmentării) bazelor de date distribuite.

În principal, va fi luată în discuție problema partiționării (fragmentării) verticale a datelor, cunoscută şi sub denumirea de partiționarea atributelor. Această tehnică se utilizează la proiectarea bazelor de date pentru îmbunătățirea performanțelor tranzacțiilor. În partiționarea verticală, atributele unei relații R sunt puse împreună în grupuri care nu se suprapun şi relația R se proiectează în fragmente de relații conform cu aceste grupuri de atribute.

În sistemele de baze de date distribuite, aceste fragmente se alocă pe site-uri diferite. De aici vine obiectivul partiționării verticale de a crea fragmente verticale ale unei relații astfel încât să se minimizeze costul accesării datelor în timpul procesului de tranzacții. Dacă fragmentele se aproprie cât mai mult de necesitățile mulțimii de tranzacții oferite, atunci costul procesului de tranzacții poate fi micşorat.

În cazul proiectării bazelor de date distribuite, costul procesului de tranzacționare se micşorează prin mărirea tranzacțiilor locale (pe un site) şi în acelaşi timp prin reducerea cantității de accesări de date care nu sunt locale. Scopul tehnicii de partiționare verticală (şi în general, a tehnicilor de partiționare) este de a găsi o schemă de partiționare care să îndeplinească obiectivul prezentat anterior.

De notat faptul că problema partiționării poate fi abordată pe diferite niveluri de detalii prin luarea în considerare a unor informații adiționale.

Fragmentarea datelor. În acestă teză, se i-au în considerare doar informațiile despre tranzacții ca dată de intrare, pentru a putea gestiona problema partiționării într-un mod eficient. De fapt, problema optimizării globale (care include un număr mare de parametri şi o metrică

complexă) se împarte în câteva probleme mai mici de optimizare pentru a putea reduce spaţiul de căutare şi a putea reduce complexitatea fiecărei probleme în parte.

În literatura de specialitate sunt propuşi câţiva algoritmi de partiţionare verticală, astfel se măsoară afinitatea dintre perechi de atribute şi se încearcă să se grupeze atributele împreună conform afinităţii între perechi de atribute, cu ajutorul algoritmului BEA (*bond energy algorithm*). Într-un articol se foloseste un estimator euristic de cost a proiectării unui fişier pentru a obţine o schemă de partiţionare "de jos-în-sus". În alt articol se extinde abordarea propusă de algoritmul BEA şi se prezintă două faze pentru partiţionarea verticală. Algoritmii de partiţionare menţionaţi anterior utilizează câteva metode euristice pentru a fragmenta o relaţie.

Ca dată de intrare a acestor algoritmi se foloseşte matricea de utilizare a atributelor (AUM). Această matrice are atribute ca şi coloane şi tranzacţii ca şi linii, iar frecvenţa accesării tranzacţiilor ca valori în matrice. Majoritatea algoritmilor anteriori folosiţi la fragmentarea datelor, utilizează matricea de afinitate a atributelor (AAM) care este derivată din AUM. AAM este o matrice în care pentru fiecare pereche de atribute, se stochează suma totală a frecvenţei de acces a tranzacţiilor care accesează acea pereche de atribute. Rezultatele diferiţilor algoritmi sunt deseori diferite, chiar dacă pentru acceaşi matrice de afinitate a atributelor, se indică faptul că funcţiile obiective folosite sunt diferite. Majoritatea algoritmilor de partiţionare verticală nu dispun de o funcţie obiectivă pentru evaluarea corectitudinii partiţiilor care se obţin în urma aplicării acelor algoritmi. De asemenea, nu există un criteriu comun sau o funcţie obiectivă care să evalueze rezultatele acestor algoritmi de partiţionare verticală.

În **Capitolul 2 „Câteva concepte de bază"** prezintă câteva noţiuni de bază referitoare la bazele de date distribuite. Astfel sunt trecute în revistă modele de reprezentare a bazelor de date relaţionale, distribuite şi orientate-obiect.

Astfel sunt descrise sistemele de baze de date în general, concentrându-se pe componentele lor software. Apoi se prezintă modelul de date relaţional şi câţiva operatori ai limbajului structurat de interogare (SQL).

Modelele de baze de date cele mai răspândite până în acest moment sunt modelul relaţional şi modelul care a reuşit să se impună în ultimii ani ca un model foarte bun, şi anume modelul orientat-obiect, sunt prezentate pe larg.

În ultimii ani, bazele de date distribuite (BDD) au devenit un sector important al prelucrării informaţiei şi se poate anticipa că importanţa lor va creşte rapid. Această tendinţă este motivată atât organizatoric cât şi tehnologic întrucât BDD elimină multe din neajunsurile BD centralizate şi sunt bine venite pentru descentralizarea structurilor organizatorice.

O BDD poate fi definită ca o colecţie de date integrate din punct de vedere logic dar fizic distribuite pe staţiile unei reţele de calculatoare.

Această definiţie pune în evidenţă două aspecte importante ale BDD:

1. Integrarea logică a colecţiilor de date. Din punct de vedere al utilizatorului există o singură bază de date cu care utilizatorul interacţionează la fel ca şi în cazul bazelor de date centralizate (BDC).

2. Distribuirea fizică se referă la partiţionarea fizică a bazei de date pe spaţii ale unei reţele de calculatoare.

Ambele aspecte sunt destul de vagi pentru a realiza deosebirile între BDD şi un set de baze de date (BD) locale. În timpul operaţiilor obişnuite, cererile de aplicaţii de la terminalele fiecărei staţii necesită numai accesul la BD locală. Aceste aplicaţii care sunt executate în întregime de computerul staţiei sunt numite **aplicaţii locale**.

Ceea ce deosebeşte un set de **BD** locale de o **BDD** este existenţa unor aplicaţii care accesează date din mai multe staţii. Aceste aplicaţii sunt numite **aplicaţii globale** sau aplicaţii distribuite. Se poate însuma consideraţiile pe care le-am făcut într-o definiţie de lucru.

O **BDD** reprezintă o colecţie de date integrate din punct de vedere logic care sunt fizic distribuite pe staţii ale unei reţele de calculatoare. Fiecare staţie a reţelei are autonomie de

prelucrare care îi permite să realizeze aplicații locale. De asemenea, fiecare stație participă la execuția aplicațiilor globale care necesită accesarea datelor din mai multe stații.

Controlul centralizat. Posibilitatea de a realiza un control centralizat asupra resurselor informaționale ale unei întreprinderi sau structuri organizatorice a fost considerată una din cele mai puternice motivații pentru introducerea BD. Funcția fundamentală a administratorului BD este de a garanta protecția şi siguranța datelor; datele în sine fiind recunoscute ca o investiție importantă a întreprinderilor care necesită un control centralizat.

În cazul BDD ideea controlului centralizat este puțin accentuată. Acest lucru depinde şi de arhitectură. În general, putem să identificăm o structură de control ierarhic bazată pe un **administrator global** care are responsabilitatea centrală a întregii BDD şi pe administratorii locali cărora le revin responsabilități legate de BD locale. Administratorii locali pot avea un grad înalt de autonomie care poate merge până la realizarea coordonării între stații. BDD diferă foarte mult în ceea ce priveşte autonomia stațiilor de la autonomia completă a stațiilor, fără existența administratorului global, până la cel mai complet control centralizat.

Independența datelor. În cazul BDD asigurarea independenței datelor față de programele de aplicații are aceeaşi importanță ca şi în cazul BDC, dar apare un nou aspect legat de **transparența distribuției**. Prin transparența distribuției programele pot fi scrise făcând abstracție de distribuirea fizică a datelor. Mutarea datelor dintr-o stație în alta afectează numai viteza de execuție, nu şi corectitudinea programului.

Ca şi în cazul independenței datelor, transparența distribuției este obținută printr-o arhitectură multinivel cu diferite descrieri ale datelor.

Asigurarea unei redundanțe minime şi controlate. În cadrul BDD există mai multe motive pentru a considera redundanța datelor o caracteristică dezirabilă:

- localizarea este mai rapidă atunci când datele sunt replicate la toate stațiile unde sunt cerute de aplicații;
- disponibilitatea, siguranța sistemului creşte atunci când datele sunt replicate. În cazul căderii unei stații, aplicațiile pot fi dirijate la stațiile unde datele sunt replicate.

Redundanța datelor reduce efortul de regăsire a datelor dar creşte efortul de actualizare. Evaluarea unui grad optim al redundanței trebuie să țină seama de raportul între accesele de regăsire şi accesele de actualizare a datelor.

Integritatea, restaurarea datelor şi controlul concurenței. În cazul BDD, realizările în domeniul integrității, restaurării datelor şi controlul concurenței, deşi se referă la probleme diferite, sunt puternic interconectate. Într-o mare măsură, soluțiile la aceste probleme sunt legate de modul de realizare a tranzacțiilor. O **tranzacție** este o unitate atomică de execuție, o secvență de operații care fie sunt realizate în întregime, fie nu sunt realizate. În cadrul BDD, problema atomicității tranzacțiilor capătă un aspect particular legat de modul în care trebuie să se comporte sistemul atunci când una din stații nu este operațională: să aboteze întreaga tranzacție sau să încerce să execute corect tranzacția chiar dacă ambele stații nu sunt simultan operaționale.

Siguranța şi securitatea datelor. În BD tradiționale, administratorul BD care are controlul centralizat permite numai un acces autorizat la date. În BDD, administratorii se confruntă cu aceleaşi probleme ca administratorii BD tradiționale. Sunt de menționat două aspecte particulare:

- în BDD, cu un grad ridicat de autonomie a stațiilor, BD locale sunt mai protejate deoarece administratorii locali îşi realizează propria protecție fără să depindă de un administrator centralizat;
- problemele securității sunt intriseci sistemelor distribuite deoarece comunicația în rețea poate reprezenta un punct slab în realizarea protecției.

În **Capitolul 3 „Arhitecturi ale bazelor de date distribuite"** sunt prezentate principalele arhitecturi ale bazelor de date distribuite, iar **capitolul 4 „Serverul se baze de date"** propune spre prezentare serverul de baze de date Oracle.

Capitolul 5 „Fragmentarea bazelor de date distribuite" este principalul capitol al acestei teze deoarece prezintă pe larg ce înseamnă fragmentarea datelor în cazul bazelor de date distribuite, pornind de la tipuri de fragmentare şi reguli de fragmentare a datelor, trecând prin prezentarea fragmentărilor orizontale, verticale şi hibride(mixte), până la utilizarea algoritmilor de grupare în fragmentarea datelor, caz particular, fragmentarea verticală atat în cazul bazelor de date distribuite relaţionale, cat si in cazul bazelor de date distribuite orientate-obiect.

Acest capitol prezintă toate noţiunile necesare înţelegerii şi aplicării procesului de fragmentare a datelor în bazele de date distribuite, atât în varianta relaţională cât şi în varianta orientată-obiect.

Astfel sunt descrise motivele pentru care se utilizează fragmentarea în procesul de proiectare a bazelor de date distribuite, împreună cu tipurile cunoscute de fragmentare, la modul general şi exemplificate cu ajutorul unei scheme de bază de date. Se prezintă gradul de fragmentare a unei baze date şi se stabilesc care sunt regulile de fragmentare care trebuiesc îndeplinite în procesul proiectării bazelor de date distribuite.

Din punct de vedere a distribuţiei datelor, nu este necesar să se fragmenteze datele. În definitiv, în sistemele de fişiere distribuite, distribuţia se execută pe baza tuturor fişierelor. În fapt, prelucrarea datelor se face cu ajutorul alocării fişierelor în nodurile unei reţele.

În fragmentare, principala caracteristică este să aproprie unităţile de distribuţie. O relaţie nu este o unitate convenabilă din mai multe motive:

- În primul rând vederile aplicaţiei sunt de obicei submulţimi de relaţii. De aceea, localizarea accesului la aplicaţii este definit nu pe toate relaţiile ci pe submulţimi ale lor. De aici este natural să considerăm submulţimile relaţiilor ca *unitate de distribuţie.*
- În al doilea rând, dacă aplicaţiile care au vederi definite pe o relaţie dată, se află pe site-uri diferite, atunci se pot urmări două alternative având întreaga relaţie ca unitate de distribuţie:
  - Pe de o parte, relaţia nu este replicată (copiată) şi este memorată pe un singur site;
  - Pe de altă parte este replicată pe toate site-urile sau pe unele dintre site-uri unde sunt aplicaţiile.

Prima variantă necesită un volum mare şi neeficient de acces controlat la date de la distanţă. A doua variantă, pe de altă parte oferă o replicare neeficientă (nenecesară), care cauzează probleme în execuţia actualizărilor şi ar putea să nu fie dorită dacă, capacitatea de memorare este limitată.

În sfârşit, descompunerea unei relaţii în fragmente, fiecare tratată ca o unitate, permite ca un număr de tranzaţii să fie executate concurenţial.

În plus, fragmentarea relaţiilor rezultă în mod tipic din execuţia în paralel a unei singure interogări prin împărţirea într-o mulţime de subinterogări care operează pe fragmente. De aici fragmentarea măreşte nivelul concurenţei.

Trebuie totuşi, amintite şi dezavantajele fragmentării. Dacă aplicaţiile au cerinţe conflictuale care previn descompunerea relaţiei în fragmente exclusive, acele aplicaţii ale căror vederi sunt definite pe mai mult de un fragment pot suferi degradări din punct de vedere al performanţei. De exemplu, ar fi necesar să recuperăm date din două fragmente şi aceasta s-ar putea obţine ori prin uniune, ori prin joncţiune, ceea ce ar fi o operaţie destul de costisitoare. Evitarea acestui aspect este o caracteristică importantă a fragmentării.

A doua problemă se referă la controlul semantic al datelor, mai ales la verificarea integrităţii datelor. Ca rezultat al fragmentării, atributele participante în dependenţe pot fi descompuse în diferite fragmente care vor fi alocate pe diferite site-uri. În acest caz, chiar şi cea mai simplă acţiune de verificare a dependenţelor se va executa prin urmărirea datelor pe un număr de site-uri.

Fragmentarea orizontală. Există două variante ale fragmentării orizontale: *fragmentare primară* şi *fragmentare derivată.*

Factorii care influenţează fragmentarea: structura bazei de date (schema conceptuală globală) şi caracteristicile aplicaţiei (predicatele folosite, locaţiile unde sunt stocate, frecvenţele de acces) – regula "80/20" (20% din interogările utilizator folosesc 80% din datele gestionate în acea bază de date).

Aplicațiile (pe care le putem considera ca fiind interogări) care accesează o bază de date, selectează o submulțime de date pe baza unui predicat – adică a unei expresii booleene.

În proiectarea bazelor de date trebuie să identificăm „grupuri" de tupluri din baza de date care sunt accesate împreună de către aplicații/interogări. Fiecare „grup" este definit ca un predicat „minterm", care este de fapt o combinație de predicate simple. Aceste „grupuri" devin apoi candidați pentru fragmentele orizontale.

Informații cantitative: *Selectivitatea minterm = sel(m_i)* (selectivitatea unui minterm m_i) şi *Frecvența de acces = acc(q_i)* (frecvența de acces a interogării q_i).

Algoritmul de partiționare orizontală este analizat şi se evidențiază paşii care trebuie parcuşi pentru a obține predicatele minterm necesare fragmentelor orizontale:

Pas 1. Construirea unei mulțimi de predicate simple care să fie completă şi minimală.

Pas 2. Generarea mulțimii de predicate minterm.

Pas 3. Eliminarea unor predicate minterm necontradictorii.

Fragmentarea orizontală derivată se defineşte pe o relație membru a unei legături în conformitate cu operația de selecție specificată pe proprietarul său.

Fragmentarea verticală. Se ştie că fragmentarea verticală a unei relații R produce fragmentele $R_1, R_2, \ldots, R_n$, fiecare conținând o submulțime de atribute ale lui R la fel ca o cheie primară a lui R.

Obiectivul fragmentării verticale este să partiționeze o relație într-o submulțime de relații mai mici astfel încât aplicațiile utilizator să ruleze numai pe un fragment.

Există două tipuri de abordări în fragmentarea verticală pe relațiile globale:

Gruparea: *începe prin asocierea fiecărui atribut unui fragment, şi la fiecare pas, unificarea fragmentelor până când criteriul este îndeplinit.*

Diviziunea (partiționarea)*: începe cu o relație şi decide care sunt partițiile utile, în conformitate cu comportamentul aplicațiilor la accesarea atributelor.*

Informațiile necesare fragmentării verticale. În faza de proiectare, fragmentarea verticală are nevoie de următoarele două informații: care aplicații/interogări folosesc atribute şi care sunt acestea frecvența cu care diferite aplicații/interogări se execută.

Prima informație va fi stocată într-o matrice numită matricea de utilizare a atributelor.
Principala informație referitoare la aplicații se referă la frecvența de acces.

Fie $Q = \{ q_1, q_2, \ldots, q_q\}$ mulțimea de interogări (aplicații) utilizator care rulează pe relația $R(A_1, A_2, \ldots, A_n)$. Atunci, pentru fiecare interogare q_i şi pentru fiecare atribut A_j, vom asocia o valoare de utilizare a atributului, notată cu use(q_i, A_j), şi definită astfel:

$$use(q_i, A_j) = \begin{cases} 1 & daca\ atributul\ A_j\ este\ utilizat\ de\ \text{int}erogarea\ q_i \\ 0 & altfel \end{cases}$$

Vectorii use(q_i, *) pentru fiecare aplicație sunt uşori de definit dacă proiectantul cunoaşte aplicațiile care vor rula pe baza de date.

Valorile de utilizare a atributelor nu sunt suficiente – în general – pentru a forma baza împărțirii atributelor şi a fragmentării. Aceasta este deoarece aceste valori nu reprezintă greutatea frecvenței aplicațiilor. Măsurarea frecvenței poate fi inclusă în definiția măsurării afinității atributelor, pe care o putem nota prin aff(A_i, A_j), care măsoară legătura între două atribute ale unei relații în conformitate cu cât sunt ele accesate de aplicații.

Măsurarea afinității dintre două atribute A_i şi A_j ale unei relații $R(A_1, A_2, \ldots, A_n)$ cu respectarea unei mulțimi de aplicații $Q = \{ q_1, q_2, \ldots, q_q\}$ este definită astfel

$$aff(A_i, A_j) = \sum_{k \,|\, use(q_k, A_i)=1 \wedge use(q_k, A_j)=1} \sum_{\forall S_l} ref_l(q_k)\, acc(q_k)$$

unde:

$ref_l(q_k)$ este numărul de accesări ale atributelor (A_i, A_j) pentru fiecare execuție a aplicației q_k pe site-ul S_l

şi $acc_l(q_k)$ este măsura frecvenţei de accesare a aplicaţiei definită mai înainte dar modificată pentru a include în definiţie frecvenţa pe site-uri diferite.

Algoritmul de grupare (clustering algorithm) aduce împreună atributele cu valori ale afinităţilor mai mari şi separat cele cu valori ale afinităţilor mai mici. Algoritmul de grupare primeşte la intrare matricea AA, permută liniile sau coloanele acesteia (este indiferent pe care le permută deoarece matricea este simetrică) şi generează matricea legăturilor de afinitate CA (clustered affinity matrix). Permutarea este făcută astfel încât să maximizeze măsura afinităţii globale (global affinity measure).

Generarea matricei legăturilor de afinitate (CA) este realizată în trei paşi, implementaţi în algoritmul GEN_CLUS_ATR:

Iniţializarea – se plasează şi se fixează o coloană arbitrară a matricei *AA* în matricea *CA* (în algoritm a fost aleasă coloana 1)

Iteraţia – se alege o coloană dintre cele *n-i* rămase (*i* este numărul coloanelor deja plasate în *CA*) şi se plasează în cele *i+1* poziţii rămase, se va alege plasarea coloanei care aduce cea mai mare contribuţie la măsura afinităţii globale descrisă mai sus. Se continuă acest pas până când nu mai sunt coloane de plasat.

Ordonarea liniilor – se schimbă poziţionarea liniilor astfel încât să fie păstrată simetria.

Algoritmul de partiţionare (partitioning algorithm) are obiectiv activitatea de partiţionare (diviziune) şi anume găsirea unei mulţimi de atribute care sunt accesate indepedent sau (în majoritatea cazurilor) de mai multe mulţimi distincte de aplicaţii.

În secţiunea 5.7 se propune o modalitate de evaluare a unei scheme de partiţionare a datelor în bazele de date distribuite relaţionale. Astfel se propune un algoritm de evaluare a fragmentelor rezultate în faza de proiectare a bazelor de date distribuite şi se analizează algoritmul pe câteva cazuri propuse.

În orice aplicaţie practică de baze de date, o tranzacţie nu necesită ca în mod obişnuit toate atributele tuplurilor unei relaţii să fie recuperate în timpul procesului unei tranzacţii.Când o relaţie este împărţită *vertical* în fragmente de date, atributelor stocate în fragmente de date care sunt irelevante (adică nu sunt accesate de tranzacţie), li se adaugă un cost de recuperare şi procesare, în special când numărul de tupluri implicate în relaţie este foarte mare.

În sistemele de gestiune a bazelor de date distribuite, când un atribut relevant (adică un atribut care este accesat într-o tranzacţie) se află în fragmente de date diferite şi care sunt alocate pe diferite site-uri, atunci se adaugă un cost adiţional pentru accesul la aceste date. De aceea una din caracteristicile dorite la sistemele de gestiune a bazelor de date distribuite pe care dorim să o atingem prin partiţionare este *accesibilitatea locală* la orice site. Cu alte cuvinte, fiecare site trebuie să fie capabil să proceseze tranzacţii *locale cu acces minim* la datele localizate pe alte site-uri.

Ideal, dorim ca orice tranzacţie să acceseze doar acele atribute dintr-un singur fragment de date fără acces deloc sau minim la atributele irelevante din acel fragment. Dar este imposibil să atingem acest caz din moment ce tranzacţiile accesează diferit şi suprapus submulţimi de atribute ale unei relaţii. Mai mult decât atât, tranzacţiile se execută pe site-uri diferite şi de aici unele dintre fragmentele de date care conţin atribute relevante ale unei tranzacţii pot fi găsite în site-uri aflate la distanţă. Costul procesului de tranzacţii într-un mediu distribuit constă din costul procesului de tranzacţii locale şi costul procesului de tranzacţii la distanţă. Chiar dacă este posibil să replicăm datele pentru a evita costul procesului, pentru primul pas presupunem că nu avem date redundante pentru a evita modelarea suprapusă şi a asigura integritatea şi consistenţa datelor şi de asemenea costul adiţional al stocării.

Presupunem că în timpul procesului de proiectare a bazelor de date, faza partiţionării este urmată de faza alocării în timpul căreia fragmente de date nesuprapuse obţinute la faza partiţionării sunt alocate pe diferite site-uri, posibil cu ceva replicare. Deci evaluatorul de partiţionare propus va evalua schemele de partiţionare verticală în care fragmentele de date sunt nesuprapuse pe atribute. Aici nesuprapuse se referă doar la atributele cheilor neprimare. Cheia primară este rezervată în fiecare partiţie. Aceasta este necesară pentru a obţine relaţia originală fără a pierde sau a adăuga tupluri.

Scopul partiționării atributelor şi alocării lor pe diferite site-uri, a ajuns la costul minim al procesării pentru orice tranzacție inițiată din orice site. Cum costul proceselor de tranzacții are două componente, una referitoare la procesarea locală şi alta referitoare la procesarea la distanță (remote), evaluatorul de partiționare propus care masoară "eficacitatea" schemei de partiționare verticală, are de asemenea doi termeni corespunzători: termenul "*costul accesului local la atributele irelevante*", şi respectiv, la termenul "*costul accesului la atributele relevante la distanță*".

Pentru a simplifica, presupunem că un singur acces la fragmente de date corespunde cu o unitate de cost; această presupunere poate cu uşurință fi relaxată dacă sunt disponibile mai multe informații referitoare la metodele de acces, parametrii rețelei, etc. Termenul de cost al accesului la atributele irelevante măsoară costul procesării locale a tranzacțiilor care se fac referitor la atributele irelevante din fragmentele de date care sunt accesate la distanță de tranzacții; de notat că această contribuție la costul de acces la distanță la atributele irelevante este deja inclusă în primul termen.

Din moment ce nu se cunoaşte în timpul partiționării cum sunt alocate fragmentele de date, se va calcula al doilea termen presupunând că fragmentele de date necesare unei tranzacții sunt localizate pe site-uri diferite. În absența oricărei informații referitoare la strategiile executării tranzacțiilor, putem calcula al doilea termen fie prin determinarea mediei aritmetice a tuturor costurilor de acces la distanță obținute prin execuția tranzacției la fiecare site ce conține un fragment necesar tranzacției sau prin presupunerea că tranzacția va rula pentru prima dată pe fragmentele site-urilor. Dacă sunt disponibile mai multe informații referitoare la strategiile de tranzacții, se pot introduce şi acestea în al doilea termen.

Secțiunea 5.8 propune, în mod asemănător cu secțiunea precedentă, o modalitate de evaluare a unei scheme de partiționare a datelor în bazele de date distribuite orientate-obiect. Algoritmul propus în secțiunea anterioară este adaptat pentru a se putea aplica în cazul bazelor de date distribuite orientate-obiect, studiindu-se câteva cazuri în funcție de metodele şi atributele specifice abordării orientate-obiect.

În abordarea relațională fragmentarea lucrează cu algoritmi care se aplică la distribuția atributelor. În funcție de tipul de fragmentare, sunt necesare informații despre frecvența tranzacției, utilizarea atributelor, sau despre tipul predicatelor. Se ştie că avem nevoie de 4 categorii de informații pentru a ajunge o proiectare optimă:

1. Informații despre baza de date care includ schema conceptuală globală.
2. Informații despre aplicație: predicate utilizate (pentru fragmentarea orizontală), sau matricea utilizării atributelor şi frecvența tranzacțiilor (pentru fragmentarea verticală).
3. Informații despre comunicarea în rețea.
4. Informații despre sistemul de calculatoare.

Totuşi, în abordarea orientată-obiect schema conceptuală poate fi mult mai complexă şi diferită de caracteristicile prezentate în abordarea relațională, aspectele adiționale trebuie să fie luate în considerare, cum ar metodele, structura ierarhică şi atributele complexe.

Aceste noi caracteristici măresc numărul de posibile abordări pentru fragmentare. De exemplu, partiționarea claselor poate implica:

- Găsirea unei "afinități" între clase (toate obiectele unei clase reprezintă o entitate aici; nu putem fragmenta atributele şi metodele). "Afinitatea" dintre clase poate fi găsită în mod similar ca la atributele din abordarea relațională.
- Unificarea claselor şi obiectelor şi găsirea "afinității" dintre atribute.
- Distribuirea metodelor folosind frecvența accesării lor prin tranzacții specifice urmate de fragmentarea atributelor corespunzătoare pe care aceste metode le accesează.
- Utilizarea frecvenței de acces a atributelor pentru partiționarea obiectelor unei clase urmată de distribuția corespunzătoare a metodelor.
- Urmărirea structurii ierarhice a claselor.

În procesul dezvoltării unui model de fragmentare, am studiat un sigur caz luat în considerare şi anume utilizarea atributelor simple şi metodelor simple.

În această categorie avem doar o ierarhie simplă, unde metodele pot folosi atributele din propria lor clasă şi/sau din superclasa lor. Nu avem metode imbricate sau obiecte ca un tip de atribut. Am folosit alte matrici de utilizare, faţă de abordarea relaţională şi anume matricea de utilizare tranzacţie-metodă – TMUM, matricea de utilizare metodă-atribut – MAUM. Matricea de utilizare tranzacţie-metodă – TMUM este matricea frecvenţelor care indică dacă tranzacţia foloseşte anumite metode. Valorile utilizate sunt zero sau unu şi reprezintă: *nu foloseşte sau foloseşte* metoda specificată. Matricea de utilizare metodă-atribut – MAUM este matricea care reprezintă numărul de invocări ale unui atribut specific într-o singură execuţie a metodei. Sunt posibile trei valori:

- zero – care indică că o metodă nu accesează un atribut.
- unu – care indică că o metodă citeste valoarea unui atribut (retrieve).
- doi – indică că o metodă citeşte şi scrie valoarea unui atribut (update).

În algoritmul propus trebuie să specificăm următoarele:

Pas 1. Să se specifice, atât în matricea TMUM cât şi în MAUM toate metodele utilizate şi atributele de la rădăcină până la nivelul ierarhic final al structurii bazei de date pe care se aplică acest algoritm. *Dacă clasele de pe nivelele mai mari decât i există, atunci pentru fiecare din aceste clase reprezentăm frecvenţa de acces a atributelor şi metodelor ca o sumă a valorilor corespunzătoare.* De precizat aici, dacă o clasă are subclase, suma va include toate frecvenţele metodelor şi atributelor acelei clase şi ale tututor subclaselor ei.

Pas 2. Înmulţim fiecare linie din TMUM cu valorile corespunzătoare ale frecvenţei de tranzacţie.

Pas 3. Înmulţim TMUM cu MAUM.

Pas 4. Aplicăm o căutare exhaustivă şi calculăm valoarea evaluatorului de partiţii pentru a selecta cea mai buna schemă de fragmentare.

Pas 5. Transmitem fragmentarea obţinută.

Scopul partiţionării atributelor este de a obţine un proces de cost minim pentru o mulţime de tranzacţii şi pentru utilizarea atributelor lor. Este puţin probabil obţinerea unei scheme de fragmentare ideală, unde fiecare tranzacţie accesează local numai atributele de care are nevoie şi nu mai are nevoie să execute nici o accesare la distanţă. Funcţia obiectivă propusă aici încearcă să echilibreze costul accesului local şi al celui la distanţă pentru o tranzacţie dată. Conform cu costul total al procesului de tranzacţii într-un mediu distribuit constă din două elemente:

1. Prima componentă, numită **costul accesului la atributele locale irelevante**, reprezintă costul accesării atributelor irelevante când se accesează un obiect într-un site local.
2. A doua componentă, numită **costul accesului atributelor relevante aflate la distanţă**, include costul accesării atributelor revelante de către o tranzacţie de la site-urile aflate la distanţă.

Capitolul 6 **„Alocarea fragmentelor”** prezintă modalităţi de alocare a datelor în bazele de date distribuite, şi propune o abordare împreună a celor două etape din proiectarea bazelor de date distribuite, şi anume fragmentarea datelor şi alocarea lor. Algoritmul prezentat este analizat împreună cu un algoritm clasic de fragmentare orizontală şi se evidenţiază, pe baza urnor evalupri experimentale, că este mai avantajos din punct de vedere al proiectării unei baze de date distribuite, să se efectueze atât fragmentarea datelor orizontal, cât şi alocarea lor pe site-urile reţelei pe care se proiectează acea bază de date distribuită.

Capitolul 7 **„Concluzii şi dezvoltări ulterioare”** prezintă, în primul rând contribuţiile originale propuse în cadrul acestei teze de doctorat, iar apoi câteva idei de dezvoltare ulterioară şi de îmbunătăţire a algoritimilor care sunt propuşi în cadrul tezei de doctorat.

Contribuţiile personale aduse de această lucrare la domeniul abordat sunt următoarele:

Prezentarea unor algoritmi clasici din domeniul fragmentării bazelor de date distribuite, în funcţie de tipurile de fragmentări de informaţii, şi anume, în cazul fragmentării orizontale primare am prezentat algoritmul de partiţionare orizontală, iar în cazul fragmentării orizontale derivate am prezentat algoritmul de fragmentare orizontală derivată.

Pe de altă parte, în cazul fragmentării verticale am prezentat algoritmii BEA (Bond Energy Algorithm) şi BVP (Binary Vertical Partitioning).

Propunerea unei modalităţi de evaluare a schemei de partiţionare a datelor într-o bază de date distribuită relaţională – algoritmul EP.

Propunerea unei modalităţi de evaluare a schemei de partiţionare a datelor într-o bază de date distribuită orientată-obiect – algoritmul EPOO.

Propunerea unei abordări euristice de fragmentare orizontală şi alocare a fragmentelor în aceeaşi fază a proiectării bazelor de date distribuite orientate-obiect. Studiul comparativ între această abordare şi un algoritm propus anterior.

Studierea performanţelor algoritmului propus într-o lucrare pe diferite date de test şi analiza rezultatelor obţinute cu evidenţierea faptului că este un algoritm util în optimizarea proiectării bazelor de date distribuite relaţionale fragmentate vertical la care nu trebuie precizat de la început numărul total de partiţii. Acest studiu a fost prezentat la International Joint Conferences on Computer, Information, Systems Sciences, and Engineering (CISSE 2007) Conference, Conference Proceedings book, 2007, University of Brigeport, USA, şi publicat în *Innovations and Advanced Techniques in Systems, Computing Sciences and Software Engineering, Springer Science*, ISBN 978-1-4020-8734-9, e-ISBN 978-1-4020-8735-6.

Studierea performanţelor algoritmului propus în altă lucrare pe diferite date de test şi analiza rezultatelor obţinute cu evidenţierea faptului că este un algoritm util în optimizarea proiectării bazelor de date distribuite orientate-obiect fragmentate vertical la care nu trebuie precizat de la început numărul total de partiţii. În această lucrare am subliniat faptul că este o adaptare a algoritmului din lucrarea anterioară, de la bazele de date distribuite relaţionale la bazele de date distribuite orientate-obiect. Acest studiu a fost prezentat şi publicat la Conferinţa 9th International Carphatian Control Conference - ICCC 2008, Sinaia, ISBN 978-973-746-897-0.

Anexele cuprind schema bazei de date distribuită folosită ca exemplificare de-a lungul întregii teze, urmată de structura bazei de date folosită în experimente, cât şi structura tabelelor componente şi de codurile sursă ale algoritmilor implementaţi în limbajul C++.

În această teză, am prezentat o abordare generală a partiţionării verticale de datelor. Acest studiu evidenţiază posibilitatea utilizării matricii de afinitate a atributelor în algoritmii de partiţionare. Folosirea unei funcţii obiective de la metodele de grupare (clustering) la algoritmii de partiţionare este prezentată printr-o implementare eficientă. Utilizând acest algoritm se pot evalua şi verifica alţi algoritmi de partiţionare verticală care folosesc ca dată de intrare matricea de utilizare a atributelor. Astfel se pot dezvolta algoritmi euristici care să ofere posibilitatea de a folosi această funcţie obiectiv de evaluare. Acest algoritm poate modificat uşor pentru include şi alte informaţii preluate de la proiectantul bazei de date, cum ar tipurile de tranzacţii (de acces la date sau de actualizare a datelor), informaţii referitoare la alocarea datelor şi costurile de transmisii de date.

Mi-am propus să dezvolt ulterior şi alţi algoritmi euristici pe baza acestei abordări propuse în lucrare şi să pot integra un astfel de algoritm într-o aplicaţie mai complexă care să ajute proiectantul bazei de date distribuite să aleagă o modalitate corectă şi cât mai eficientă de implementare a informaţiilor gestionate cu ajutorul unei baze de date distribuite.