# UNELE ASPECTE DE PROIECTARE A BAZELOR DE DATE DISTRIBUITE

***Dr.conf.univ.V. Cotelea,***
*Academia de Studii Economice din Moldova*

O bază de date, proiectată corespunzător, furnizează acces la informaţii precise, actualizate. Deoarece o proiectare corectă este esenţială pentru atingerea scopurilor utilizării unei baze de date, capacitatea de a proiecta baze de date şi aplicaţii asociate este critică pentru succesul oricărei întreprinderii moderne. Elaborarea automată şi dezvoltarea bazei de date şi a sistemului informaţional, în întregime, constă dintr-o serie de etape, care necesită cercetări metodologice, modele şi algoritmi eficienţi şi instrumente software de proiectare.

## 1. STRATEGII DE PROIECTARE A BAZELOR DE DATE

Proiectarea bazei de date integrate pentru un domeniu de interes complex şi, de obicei, de dimensiuni mari este o sarcină grea. În abordarea problematicii de proiectare a bazelor de date, se poate opta, în general, pentru două tipuri de strategii: strategia ascendentă şi strategia descendentă [7]. Ambele tipuri nu sunt mutual exclusive şi, la diferite etape ale proiectării, poate fi aplicată o strategie sau alta.

Strategia ascendentă se aplică în cazul când proiectarea are loc, pornind de la un număr de baze de date existente mai mici, cu un sfârşit de integrare într-una. În această strategie, se porneşte de la schemele conceptuale locale şi se lucrează pentru construirea schemei conceptuale globale. Cu toate că acest caz se poate realiza uşor în viaţa reală, se preferă cazul când se porneşte de la zero şi se lucrează, folosind strategia descendentă. Strategia descendentă se realizează, de obicei, de o persoană (sau un grup de persoane) care posedă cunoştinţe despre tehnicile proiectării bazei de date, cu excepţia fazei de proiectare a distribuţiei (în cazul bazelor de date distribuite [30]).

Obiectivele principale [18] ale proiectării bazei de date sunt următoarele:

- Satisfacerea cerinţelor privitoare la conţinutul de date şi aplicaţii;
- Oferirea unei structuri simple şi comprehensive de date;
- Susţinerea cerinţelor de prelucrare (timpul de răspuns, timpul de prelucrare, spaţiul de stocare...);
- Oferirea unei scheme flexibile a bazei de date, adică uşor modificabile (ca urmare a schimbării cerinţelor).

Cel mai frecvent este utilizată strategia descendentă. Motivul constă în faptul că această strategie asigură mai bine elaborarea instrumentelor integrate, pentru a ajuta atât proiectarea bazelor de date, cât şi proiectarea aplicaţiilor care utilizează baza de date [21].

Organizarea datelor pentru a satisface aceste cerinţe începe cu studierea detaliată a aplicaţiilor bazei de date şi semanticii acestora. Pentru a simplifica dificultatea, complexitatea şi sarcinile consumatoare de timp (cronofage), procesul de proiectare, de obicei, este divizat în patru faze mari [34]: analiza cerinţelor informaţionale, proiectarea conceptuală, proiectarea logică şi proiectarea fizică.

Totul începe cu o analiză a cerinţelor, care definesc domeniul de interes, pentru a obţine toate necesităţile de prelucrare a datelor posibililor utilizatori ai bazei de date. De asemenea, pot fi fixate cerinţele sistemului, obiectivele care trebuie atinse, ţinând cont de un grad înalt de eficienţă, securitate [16], disponibilitate şi flexibilitate, fără a uita importantul aspect economic.

Rezultatele acestui pas servesc drept intrare pentru două activităţi care se realizează paralel [8]. Proiectarea viziunilor, pe de o parte, constă în definirea interfeţelor pentru utilizatorii finali, iar pe de altă parte, proiectarea conceptuală necesită examinarea domeniului de interes pentru a determina entităţile şi asocierile dintre ele. Există, totuşi, o legătură strânsă între proiectarea viziunilor şi proiectarea conceptuală. De obicei, limbajul folosit atât pentru modelarea viziunilor, cât şi pentru proiectarea conceptuală, în întregime, se foloseşte modelul Entitate-Asociere [12, 14] sau limbajul UML [26].

Proiectarea conceptuală poate fi interpretată, de asemenea, ca integrarea viziunilor utilizatorilor. Acest aspect este de importanţă vitală, deoarece modelul conceptual trebuie să suporte nu numai aplicaţiile existente, dar şi aplicaţiile viitoare. În proiectarea conceptuală şi a viziunilor, se specifică entităţile de date şi se determină aplicaţiile care vor

funcţiona asupra bazei de date pentru a se extrage date statistice sau estimaţii asupra activităţilor acestor aplicaţii. Ele vor include date despre frecvenţele de acces ale aplicaţiilor la relaţii distincte şi despre atributele schemelor la care aplicaţiile acced. Rezultatul lucrului efectuat până aici rezidă în schema conceptuală globală calitativă [10], care este transformată în schemă logică globală [15]. Deoarece schema conceptuală, deseori, este reprezentată în formă grafică, apare necesitatea soluţionării problemei de minimizare a intersecţiilor [9].

Astfel, la această etapă de proiectare este nevoie de tehnici şi instrumente pentru:

- Selectarea datelor despre domeniul de interes, analiza necesităţilor informaţionale, determinarea proceselor necesare pentru realizarea funcţiilor potenţialilor utilizatori din punct de vedere a structurii informaţiei.
- Reprezentarea modelului semantic pentru fixarea datelor despre domeniul de interes.
- Analiza datelor acumulate, clasificarea, formalizarea integrarea elementelor structurale, formularea constrângerilor de integritate atât structurale, cât şi procedurale, precum şi determinarea dinamicii obiectelor din domeniul de interes.
- Sinteza modelului conceptual în limbajul semantic selectat cu acest scop.

Din păcate această fază de proiectare este greu formalizabilă, deci şi greu automatizabilă. Acest lucru se explică şi prin faptul că înţelegerea domeniului de interes, mai ales a semanticii, aşa cum este concepută de fiecare potenţial utilizator, este o problemă cognitivă ce ţine de domeniul inteligenţei artificiale. Singura soluţie, acum, constă în elaborarea unor sisteme CASE (Computer Aided System Engineering) pentru rezolvarea unor probleme concrete de susţinere a limbajelor semantice şi susţinerea unor documente de proiectare conceptuală [32].

La faza de proiectare logică, schema conceptuală, exprimată într-un model de date de nivel înalt, este transformată în schemă logică globală descrisă într-un model de date logic, de exemplu, modelul relaţional, fără a se ţine cont de un SGBD concret; în acest caz, se obţine un proiect logic independent de sistem, dar dependent de modelul de date.

Schema logică globală se normalizează [5, 6], sunt identificate toate cheile [23], legăturile dintre relaţii. Apoi schema logică globală şi informaţia despre accesul la date servesc drept intrare pentru pasul următor, proiectarea distribuţiei [30]. Obiectivul acestei etape constă în proiectarea schemelor logice locale, care se distribuie tuturor posturilor sistemului distribuit. În acest caz, este posibilă tratarea fiecărei entităţi ca unitate de distribuţie, or, în cazul modelului relaţional, orice entitate corespunde unei relaţii. În consecinţă, frecvent, orice relaţie se împarte în subrelaţii mai mici, numite fragmente, care apoi se plasează pe unul sau mai multe calculatoare. Astfel, procesul de proiectare a distribuţiei constă din două activităţi principale: fragmentarea şi alocarea.

Ultima fază este proiectarea fizică care, printre altele, distribuie schemele logice locale în conformitate cu dispozitivele de memorare fizică disponibile pe diverse calculatoare. Intrările pentru acest pas sunt schemele logice locale şi informaţiile de acces la fragmente. În sfârşit, se ştie că activitatea de elaborare şi proiectare constituie un proces ce necesită o monitorizare şi o ajustare periodică. Astfel, dacă se produc devieri, se poate reveni la faza respectivă anterioară.

Proiectarea fizică reprezintă un proces de alegere a structurilor de stocare pe disc şi metodelor de acces adecvate pentru obţinerea unei exploatări eficiente a bazei de date [11]. În momentul construirii schemei fizice, este important să se cunoască sarcinile de lucru (consultările şi actualizările) ce trebuie să le asigure baza de date şi cerinţele referitoare la performanţele înainte de către utilizatori. Este important, de asemenea, să fie cunoscute tehnicile de procesare a interogărilor şi indexare [13], susţinute de SGBD, alte caracteristici ale SGBD-ului, sistemului de operare şi echipamentului pentru a atinge următoarele obiective:

- Diminuarea timpului de răspuns la interogări;
- Minimizarea spaţiului de stocare a datelor;
- Evitarea reorganizării fişierelor;
- Asigurarea unei securităţi maximale;
- Optimizarea consumului de resurse.

## 2. PROBLEMATICA FRAGMENTĂRII VERTICALE A BAZEI DE DATE

O bază de date distribuită ar trebui să apară utilizatorilor exact ca o bază de date nedistribuită [17]. Datele de pe fiecare calculator sunt gestionate independent de către SGBD, care este responsabil pentru menţinerea integrităţii lor.

Proiectarea bazelor de date distribuite reprezintă o activitate complexă, ce trebuie să ia în considerare o mulţime de factori, precum cerinţele de accesare ale aplicaţiilor, restricţiile de integritate definite la nivelul întregii baze de date, configuraţia nodurilor din sistem, precum şi a reţelei. Natura

distribuită a datelor implică, pe lângă activitățile care privesc proiectarea bazelor de date centralizate, alte două: fragmentarea şi alocarea.

Motivele de fragmentare a unei relații sunt: uzanța, eficienta, paralelismul şi securitatea [29].

Referitor la uzanță, trebuie menționat că aplicațiile lucrează mai bine cu viziuni , decât cu relații întregi. Prin urmare, pentru distribuirea datelor, pare mai adecvat să se lucreze cu submulțimi ale relațiilor ca unități de distribuire.

Eficiența este mai înaltă datorită faptului că datele sunt stocate in apropierea locului unde sunt cel mai frecvent utilizate, iar datele care nu sunt necesare pentru aplicațiile locale nu sunt stocate.

Fragmentele fiind unități de distribuire, tranzacția poate fi împărțită în mai multe subinterogări, care operează asupra fragmentelor. Aceasta are ca efect mărirea concurenței din sistem, permițând executarea în siguranță a tranzacțiilor care pot fi executate în paralel.

Securitatea se explică prin faptul că datele care nu sunt necesare pentru aplicațiile locale nu sunt stocate şi, in consecință, nu sunt disponibile pentru utilizatorii neautorizați.

Soluționarea problemei fragmentării trebuie să se găsească în strânsă legătură cu defragmentarea. Defragmentarea completă sau parțială este, uneori, necesară în cazul când performantele aplicațiilor care utilizează date din mai multe fragmente, aflate pe stații diferite, care pot fi scăzute. În afară de aceasta, controlul integrității poate fi mai dificil, dacă datele şi dependențele funcționale sunt fragmentate în stații diferite.

Un fragment vertical al unei relații este format dintr-o mulțime de atribute ale acesteia. Prin fragmentarea verticală, se grupează laolaltă atributele care sunt utilizate de către unele aplicații. Se definește cu ajutorul operației de proiecție din algebra relațională.

Fragmentarea, în general, precum şi fragmentarea verticală, în particular, nu poate fi efectuată la întâmplare. În decursul fragmentării, este necesar să fie urmate trei reguli. La fel ca şi alte tipuri de fragmentări, cea verticală trebuie să respecte caracterul complet, care garantează că nu au loc pierderi de date în decursul fragmentarii, adică fiecare atribut al relației inițiale trebuie să facă parte cel puțin dintr-un fragment.

În afară de aceasta, fragmentarea verticală trebuie să păstreze constrângerile de integritate, fiind reconstruibilă. Adică, trebuie să fie posibil să se definească o expresie relațională, care va reconstrui relația inițială din fragmente.

În ceea ce privește caracterul disjunct al fragmentării, fragmentarea verticală reprezintă o excepție de la această regulă, în care atributele cheii primare trebuie repetate pentru a permite reconstrucția relației inițiale. Prin această regulă, se garantează redundanța minimă a datelor.

Dar când baza de date proiectată trebuie să fie distribuită? Există multe motive pentru care pot impune proiectarea şi folosirea unei baze de date distribuite. Cele mai importante dintre acestea, în opinia [30], sunt următoarele:

- Necesitatea de a plasa datele accesate frecvent în apropierea aplicațiilor-client, care au nevoie de acestea şi reduce astfel numărul de mesaje în rețea şi timpul de acces.
- Dorința de a plasa datele variabile într-un singur loc, astfel, reducându-se la minim problemele asociate cu prezența mai multor exemplare actualizate ale datelor.
- Necesitatea de a reduce impactul, cum ar fi căderea unui singur server, pe care se găsește baza de date server.

Dacă aceşti factori sunt esențiali pentru funcționarea cu succes a aplicațiilor ce trebuie elaborate pentru un domeniu de interes, bazele de date distribuite pot fi exact lucrul de care este nevoie. Totuşi, proiectarea unei baze de date distribuite, întreținerea acesteia şi după punerea în funcțiune nu pot fi numite triviale.

Există mai multe moduri de a organiza distribuirea datelor.

Hoffer şi Severance au introdus conceptul de afinitate a atributelor [20]. Acest concept măsoară frecvența de accesare simultană a unei perechi de atribute. Atributele, având afinitate mare, sunt grupate împreună folosind algoritmul BEA (Bond Energy Algorithm) elaborat de McCormick şi alții [25].

Hammer şi Niamir [19] au propus o euristică în cazul în care intrarea reprezintă un set de blocuri care corespunde fiecare unui atribut. Aceasta este partiția inițială. La fiecare pas, sunt generate o serie de modificări ale partițiilor şi apoi sunt depuse la un evaluator de cost. Dacă una din partițiile modificate devine partiția-candidat curentă, iar procesul de căutare continuă până când nu este posibilă nici o modificare. Modificarea unei partiții poate fi obținută prin două moduri diferite: prin gruparea a două blocuri sau prin extragerea unui atribut dintr-un bloc şi introducerea acestuia în altul.

În literatura de specialitate, sunt propuşi câțiva algoritmi de fragmentare verticală. Se măsoară afinitatea dintre perechi de atribute şi se încearcă să se grupeze atributele împreună conform afinității între perechi de atribute, cu ajutorul algoritmului BEA. Astfel, Navathe şi alții [29] au extins lucrarea [20], aplicând matricea de afinitate a

atributelor şi algoritmul BEA. Algoritmul de fragmentare presupune două etape. La prima etapă, fragmentarea este obţinută prin aplicarea iterativă a unui algoritm de partiţionare binar. La acest pas, nu este considerat factorul de cost. La a doua etapă, estimările costului, care reflectă o schimbare a mediului fizic, sunt incluse cu scopul de a optimiza fragmentele iniţiale. Complexitatea algoritmului este $O(n^2 \log n)$, unde $n$ este numărul de atribute.

Majoritatea algoritmilor utilizează matricea de afinitate a atributelor care este derivată din matricea de utilizare a atributelor. Această matrice are atribute ca şi coloane şi tranzacţii ca şi linii, iar frecvenţa accesării tranzacţiilor ca valori în matrice.

Cornell şi Yu [Corn 1987] au propus un algoritm de partiţionare verticală care minimizează numărul de accesări ale discului. Algoritmul se bazează pe metodele de programare cu numere întregi. Partiţionarea unei relaţii necesită cunoaşterea mai multor parametri în ceea ce priveşte relaţia (lungimea, selectivitatea şi numărul de atribute), precum şi tipurile de tranzacţii şi comportamentul acestora (frecvenţa şi atributele accesate).

Ceri şi alţii [3] propun două instrumente pentru fragmentarea verticală: "*Divide*" si "*Conquer*". Instrumentul "Divide" realizează numai fragmentarea şi alocarea datelor şi pune în aplicare algoritmul de partiţionare propus în [29]. Instrumentul "*Conquer*", în afară de fragmentarea şi alocarea datelor, asigură optimizarea şi alocarea operaţiunilor.

Navathe şi Ra au îmbunătăţit lucrarea anterioară privind fragmentarea verticală prin propunerea unui algoritm folosind o tehnică grafică [27]. Matricea de afinitate a atributelor este considerată ca un graf complet, unde nodurile reprezintă atributele, iar ponderile muchiilor reprezintă valorile de afinitate. Algoritmul, prin adăugarea succesivă a muchiilor, generează toate fragmentele într-o singură iteraţie, considerând un ciclu drept un fragment. Algoritmul are o complexitate de $O(n^2)$, unde $n$ este numărul de atribute şi are avantajul că nu utilizează o funcţie obiectiv.

Lin şi alţii [22] extind rezultatele din [27] despre partiţionarea grafică. În calitate de date de intrare ale algoritmului, figurează graful de afinitate a atributelor. Autorii au propus căutarea unui subgraf cu cel puţin două noduri pentru care valorile de afinitate sunt mai mari decât cele de la fiecare muchie incidentă.

Chakravarthy şi alţii [4] au prezentat un evaluator al partiţiei pentru a măsura calitatea unei fragmentări verticale prin utilizarea a două costuri: costul de acces la atributele locale irelevante (prezente pe staţia de executare a tranzacţiei, dar care nu sunt utilizate de tranzacţia în cauză) şi costul de acces la atributele îndepărtate irelevante (care nu sunt prezente pe staţia de executare a tranzacţiei, dar necesare pentru executarea acesteia).

Navathe şi alţii [28] au propus o tehnică de fragmentare mixtă. Datele de intrare ale algoritmului sunt un tabel de afinitate a predicatelor şi un tabel de afinitate a atributelor. Ma şi alţii [24] au folosit o matrice de frecvenţă a utilizării atributelor şi un model de cost pentru fragmentarea verticală.

Abdalla şi alţii [1] au utilizat matricea de afinitate a atributelor pentru a genera grupuri bazate pe valorile de afinitate.

Abuelyaman [2] a propus un algoritm static, StatPart, pentru fragmentarea verticală. În această lucrare, a fost furnizată o soluţie pentru fragmentarea iniţială a relaţiilor într-o bază de date distribuită. O matrice de reflexivitate, o matrice de simetrie şi un modul de tranzitivitate, generate în mod aleatoriu, au fost folosite pentru a produce fragmente verticale ale relaţiilor şi nu pentru fragmentarea orizontală. Din păcate, nu poate fi justifică ipoteza că tehnica propusă va produce o fragmentare bună.

Rezultatele diferiţilor algoritmi sunt, deseori, diferite, chiar dacă, pentru aceeaşi matrice de afinitate a atributelor, se indică faptul că funcţiile obiective folosite sunt diferite. Majoritatea algoritmilor de fragmentare vertical nu dispun de o funcţie-obiectiv pentru evaluarea corectitudinii partiţiilor care se obţin în urma aplicării acelor algoritmi. De asemenea, nu există vreun criteriu comun sau o funcţie-obiectiv care să evalueze rezultatele acestor algoritmi de partiţionare verticală.

Dacă fragmentarea orizontală a schemei logice globale este o problemă rezolvabilă [30] într-un timp acceptabil, atunci pentru fragmentarea verticală şi mixtă nu poate fi argumentat vreun algoritm că se obţine o soluţie bună. Prin urmare, este necesară utilizarea tehnicilor inteligente. Cu toate că metodele inteligenţei artificiale, rareori, schimbă natura problemei din punctul de vedere al complexităţii temporale şi nu produc soluţii optimale, acestea aduc, într-un timp acceptabil, o soluţie bună, aproape de cea optimală.

Îndată ce schema logică globală este fragmentată şi alocată, apar problemele ce trebuie soluţionate la faza de proiectare logică pe fiecare staţie aparte, care, de fapt, nu se deosebesc, în principiu, de problemele de proiectare logică în sistemele centralizate sau cu arhitectură client server [7].

## 3. CORELAȚIA DINTRE FAZELE LOGICĂ ŞI FIZICĂ DE PROIECTARE

În mod tradițional, activitățile de proiectare a bazei de date sunt împărțite în faze distincte, în care faza de proiectare logică precede proiectarea fizică a bazelor de date. [7]. Obiectivul proiectării logice, printre altele, constă în eliminarea redundanțelor şi anomaliilor de actualizare, apelând la dependențe de date, lăsând în acelaşi timp, fazei de proiectare fizică identificarea modului în care schema bazei de date pot fi restructurate pentru a oferi un acces mai eficient la date.

Recunoscând că normalizarea, deseori, provoacă joncțiuni scumpe în procesarea interogărilor, mulți practicieni recurg la denormalizare în calitate de activitate integrală în proiectarea fizică a bazelor de date [31].

Denormalizarea este procesul prin care, după definirea unei scheme deplin normalizate, se combină relațiile pentru a mări viteza de acces, evitând o serie de operații de joncțiune.

Unii cercetători leagă denormalizarea strict de nivelul fizic al schemei, privind-o ca înregistrări fizice nenormalizate. Alții consideră că procesul denormalizării se desfăşoară, preponderent, la nivel logic. Firește, în această lucrare se tratează aspectele ce țin mai mult de schemele logice şi, în mai mică măsură, fizice ale bazei de date.

Date [17] desface argumentele denormalizării de mai sus, deoarece acestea pornesc de la premisa falsă că o relație corespunde unui fişier de bază fizic. Devierea de la normalizare poate fi aplicată dacă alte mecanisme de optimizare nu sunt adecvate sau disponibile, sau cerințele de performanță sunt absolut necesare, sau există mecanisme de asigurare a integrității şi consistenței bazei de date. Trebuie menționat că denormalizarea are, totuşi, un cost, care se exprimă în pierderea flexibilității, micşorarea scalabilității, precum şi mecanisme de păstrare a corectitudinii datelor în cazul redundanței.

Metodele de denormalizare propuse nu iau, însă, în considerare semantica datelor implicate şi se pierde integritatea datelor şi, deseori, respectarea integrității este lăsată pe umerii programatorilor de aplicații în regim ad-hoc.

Se presupune că fazele de proiectare fizică şi logică nu trebuie să fie activități disjuncte. Se susține că separarea acestor două etape, de multe ori, rezultatul nu beneficiază de cunoştințele de semantică a datelor capturate la fazele anterioare ale ciclului de proiectare a bazei de date.

Un pas spre depăşirea acestei probleme constă în extinderea noțiunii clasice de dependență funcțională pentru a capta semantica datelor relevante şi a asigura o alegere mai inteligentă pe parcursul proiectării fizice, oferind reguli pentru o proiectare mai eficientă a schemelor atâta timp cât nu este compromisă integritatea bazei de date. Acest lucru este realizat prin introducerea dependențelor funcționale puternice şi slabe.

Dependențele funcționale puternice indică faptul că legătura dintre două mulțimi de atribute nu se schimbă aproape niciodată. Acest concept permite să se controleze redundanțele, ceea ce este benefic, deoarece poate reduce semnificativ efortul necesar pentru a accesa frecvent informațiile necesare.

Dependențe funcționale slabe capturează situațiile comune din viața reală în cazul în care dependențe funcționale clasice între două mulțimi de atribute se respectă, în general, dar pot fi încălcate, în cazuri rare.

Astfel, în lucrarea [33], se arată că constrângerile impuse de forma normală trei pot fi relaxate pentru a facilita proiectarea schemei bazei de date care ar fi mai eficientă din punctul de vedere al procesării şi stocării. Aceasta se realizează prin identificarea acelor circumstanțe sub care pot fi permise unele relaxări fără sacrificarea integrității sau performanței bazei de date. Astfel s-a ajuns la trei forme normale noi: forma normală trei relaxată, forma normală trei replicată şi forma normală trei relax-replicată. Formele normale sunt definite, utilizând dependențele funcționale slabe şi puternice, care oferă un cadru teoretic pentru proiectarea schemele de baze de date, şi sunt mai eficiente şi practice, şi, în acelaşi timp, nu compromit integritatea bazei de date. Adică aceste legături nu vor suferi de anomalii de actualizare indezirabile.

În [33] sunt prezentate două tehnici de proiectare schemei în formele normală trei relaxată şi replicată. De asemenea, sunt prepuse metode de menținere a integrității bazei de date în respectivele forme normale. În mod analogic, pot fi definite forme normale Boyce-Codd noi.

Trebuie menționat că, deseori, proiectanții cad în capcana confuziei dintre denormalizare şi redundanță. Este nevoie să se țină cont de gradul de accesare a diferitelor atribute sau grupuri de atribute şi denormalizarea se cere evitată dacă datele îmbinate prezintă diferențe semnificative.

## 4. CONCLUZII

Astăzi, multe aplicaţii cer ca bazele de date să evolueze independent de intervenţia utilizatorului, ci ca răspuns la un eveniment sau la o situaţie determinată. În sistemele de gestiune a bazelor de date tradiţionale (pasive), evoluţia bazei de date se programează în codul aplicaţiilor, în timp ce, în sistemele de gestiune a bazelor de date active, această evoluţie este autonomă şi se defineşte în schema bazei de date. Astfel, schemele bazelor de date trebuie proiectate cu caracteristici care pot evolua, în mod autonom, în funcţie de influenţa mediului constituit din aplicaţii şi interacţiuni ale utilizatorilor. Prin sistemele de baze de date active, se constituie un nou nivel de independenţă a datelor: independenţa cunoştinţelor.

Din această perspectivă, provocarea pentru cercetătorii din domeniul bazelor de date constă în a identifica şi a unifica cadrul teoretic, care ar integra diferite faze ale proiectării bazelor de date într-un mod coerent

Privitor la soluţionarea problemei fragmentării, se poate constata că: majoritatea algoritmilor cunoscuţi de fragmentare verticală nu dispun de o funcţie-obiectiv pentru evaluarea corectitudinii partiţiilor; nu există vreun criteriu comun sau o funcţie-obiectiv, care să evalueze rezultatele acestor algoritmi de partiţionare; rezultatul diferiţilor algoritmi sunt, deseori, diferite, chiar dacă pentru aceeaşi matrice de afinitate a atributelor se indică faptul că funcţiile-obiectiv folosite sunt diferite; nu pentru toţi algoritmii poate fi justifică ipoteza că tehnica propusă va produce o fragmentare bună. Ţinând cont de faptul că, în caz general, problema fragmentării verticale este una exponenţială, sunt necesare cercetări orientate spre utilizarea tehnicilor inteligente şi anume algoritmii genetici.

Aceste tehnici trebuie să ofere soluţii acceptabile, aproape de cele optimale, în timp rapid. Cu acest scop sunt necesare investigaţii privind reprezentarea indivizilor şi crearea populaţiei iniţiale, determinarea gradului de adaptare a indivizilor, aplicarea operatorilor şi parametrilor genetici şi regulile de utilizare a acestora. Sunt necesare, de asemenea, experimente asupra unor exemple mici, pentru care pot fi calculate toate variantele posibile de fragmentare, şi să se arate că algoritmul produce soluţii care se situează printre cele care respectă pragul de vigilenţă.

Activităţile de proiectare a bazei de date sunt grupate în faze distincte, care utilizează, deseori, modelele proprii de date, tehnicile particulare etc. Cu toate acestea, separarea absolută a fazelor de proiectare, deseori, este susţinută de intervenţia utilizatorului în ceea ce priveşte introducerea datelor necesare la fiecare pas. Crearea unor sisteme viabile presupune neapărat utilizarea paradigmei de independenţă (sau dependenţă) relativă a diferiţilor paşi de proiectare. Astfel, problema de corelare a fazelor necesită investigaţii complexe cu privire la modelele de date şi tehnicile care capturează semantica datelor obţinute la fazele anterioare ale procesului de proiectare, fapt care va fi neglijat în cazul activităţilor disjuncte.

### *Bibliografie*


1. ***Abdalla H., AlFares M., Marir F.*** *Vertical Partitioning for Database Design: A Grouping Algorithm. In Proc. International Conference on Software Engineering and Data Engineering (SEDE), 2007, p. 218-223.*
2. ***Abuelyaman E. S.*** *An optimized scheme for vertical partitioning of a distributed database. Int. Journal of Computer Science & Network Security, V.8, N.1, 2008, p.310-315.*
3. ***Ceri S., Pernici S., Weiderhold G.*** *Optimization Problems and Solution Methods in the Design of Data distribution. Information Sciences, V.14 N.3, 1989, p.261-272.*
4. ***Chakravarthy S., Muthuraj J., Varadarajan R., Navathe S. B.*** *An objective function for vertically partitioning relations in distributed databases and its analysis. Distributed and Parallel Databases, Springer, V.2, N.2, 1994, p.183–207.*
5. ***Codd E.F.*** *Further Normalization of the Database Relational Model. In Data Base Systems, Courent Comp.Sci. Symposia Series, 6, Englewood Cliffs, NJ: Prentice-Hall, Rustin, 1972, p.33-64.*
6. ***Codd E.F.*** *Recent Investigation in Relation Data Base Systems. IFIP Congress, 1974, p.1017-1021.*
7. ***Connolly Thomas M., Begg Carolyn E.*** *Database systems: a practical approach to design, implementation, and management. Fourth Edition, Addison-Wesley, 2005, 1374 p.*
8. ***Cotelea V., Caranicolov A.*** *Sistem automatizat de proiectare conceptuală şi logică a bazelor de date. Bucureşti, 28-29 noiembrie 1991- Tezele a V-a conferinţă Naţională de Cibernetică.p.17-19.*
9. ***Cotelea V., Pripa S.*** *An algorithm of graph planarity testing and cross minimization. Computer Science Journal of Moldova. V.15, N.3 (45), 2007, p.278-287.*
10. ***Cotelea V.*** *Calitatea schemei conceptuale a bazei de date în modelul entitate-asociere. Analele Academiei de Studii Economice din Moldova, Chişinău, Editura ASEM, 2004, p.428-432.*

*11.* ***Cotelea V.*** *Capitolul 10. Structuri de date şi acces în baze de date: în Structuri de Date, Editura ASE, Bucureşti, 2008, Volumul II, p.236-306.*
*12.* ***Cotelea V.*** *Inconsistenţe în modelul Entitate-Relaţie. Simpozionul ştiinţific "Formarea economiei eficiente prin forţele pieţei", Chişinău, ASE, 1995, p.83-87.]*
*13.* ***Cotelea V.*** *Indecşi pentru interogarea ierarhiilor de moştenire ale claselor în bazele de date orientate pe obiecte. Drept, economie şi informatică, N.2(8), 2005, Galaţi, p.124-130.*
*14.* ***Cotelea V.*** *Modelul Entitate-Relaţie: de la semantică la proiectare. Economica, An 3, Nr.1(6), ASE, Chişinău, 1995, p.153-160.*
*15.* ***Cotelea V.*** *Principii de proiectare conceptuală interactivă a bazelor de date. Cibernetica şi informatica economică. A.S.E., Chişinău, 1996, p.127-131.*
*16.* ***Cotelea V.*** *Tehnici şi modele în securitatea bazelor de date. Com. Simpozionul ştiinţific "Politica industrială şi comercială în Republica Moldova, Chişinău, 25-26 septembrie 1997, Editura ASEM, p.352-357.*
*17.* ***Date C. J., Darwen Hugh.*** *Databases, Types, and the Relational Model. The Third Manifesto. Addison Wesley; 3th edition, 2006, 572 p.*
*18.* ***Elmasri Ramez, Navathe Shamkant B****. Fundamentals of Database Systems. Addison-Wesley Publishing Company, 5th edition, 2006, p.1168.*
*19.* ***Hammer N. Niamir B.*** *A heuristic approach to attribute partitioning. In Proceedings ACM SIGMOD Int. Conf. on Management of Data, (Boston, Mass.), ACM, New York. p. 93-101.*
*20.* ***Hoffer J., Severance D.*** *The Uses of Cluster Analysis in Physical Database Design. In Proc. 1st International Conference on VLDB. Framingham, MA, 1975, p. 69-86.*
*21.* ***Hoffer J.A., Prescott M.B., McFadden F.R.*** *Modern Database Management. 9th Edn., Pearson Education Inc-Prentice-Hall, 2009, 736 p.*
*22.* ***Lin X., Orlowska M., Zhang Y.*** *A Graph Based Cluster Approach for Vertical Partitioning in Database Design. Data and Knowlegde Engineering, V.11, N2, 1993, p. 151-169.*
*23.* ***Lucchesi C.L., Osborn S.L.*** *Candidate keys for relations.- Jour. Of Comput. And Syst. Sci., 1978, N.17, p.270-279.*
*24.* ***Ma H. Schewe K. D, Kirchberg M.*** *A heuristic approach to vertical fragmentation incorporating query information. In Proc. 7th International Baltic Conference on Databases and Information Systems (DB&IS), 2006, p.69–76.*
*25.* ***McCormick W.T., Schweitzer P.J., White T.W.*** *Problem decomposition and data reorganization by a clustering technique. Operations Research, V.20, 1972, p. 993–1009.*
*26.* ***Naiburg Eric J., Maksimchuk Robert A.*** *UML for Database Design. Addison-Wesley Professional, 2001, 320 p.*
*27.* ***Navathe S. B., Ra M.*** *Vertical partitioning for database design: A graphical Algorithm. ACM SIGMOD Record, V.14, N.4, 1989, p.440-450.*
*28.* ***Navathe S., Karlapalem K., Ra M.*** *A mixed fragmentation methodology for initial distributed database design. Journal of Computer and Software Engineering, V.3, N.4, 1995, p 395–426.*
*29. Navathe Shamkant, Ceri Stefanol Wiederhold Gio, Dou Ju. Vertical Partitioning Algorithms for Database Design, ACM Trans. on Database Systems, V.9, N.4, 1984, p. 680-710.*
*30.* ***Özsu M.T., Valduriez P.*** *Principles of Distributed Database Systems, ed. Dorling Kindersley (india) Pvt Ltd, 2006, 720 p.*
*31.* ***Silberschatz A., Korth H.F., Sudarshan S****. Database System Concepts, Sixth Edition, McGraw-Hill, 2010, 1376 p.*
*32.* ***Ullman Jeffrey D.*** *Principles of Database and Knowledge-Base Systems Vol. II: The New Technologies. Spektrum Akademischer Verlag, 1990, 511 p.*
*33.* ***Wang Ling Tok, Hian Goh Cheng, Li Lee Mong.*** *Extending classical functional dependencies for physical database design. Information and Software Technology, V.38, N.9, 1996, p.601–608.*
*34.* ***Yao S. Bing.*** *Principles of Database Design, Vol.I: Logical Organizations. Prentice-Hall, 1985, p.496.*

***Recomandat spre publicare: 29.01.2012.***