O PRIVIRE INTUITIVĂ ASUPRA MODELULUI DE DATE ORIENTAT PE OBIECTE COMPLEXE

Mat. Mircea Râureanu,
Ec. Virgil Duţescu

Institutul de Cercetări în Informatică

REZUMAT

Lucrarea îşi propune să studieze modelul de date, orientat pe obiecte complexe din perspectiva bazelor de date relaţionale.

Se poate observa că, dacă se încalcă restricţia 1NF dată de teoria normalizării, se obţine un concept interesant: **relaţia structurată.**

Evidenta similitudine între relaţiile imbricate şi obiectele complexe a determinat ideea extinderii algebrei relaţionale în scopul realizării de SGBD-uri orientate pe relaţii structurate. În lucrare sînt definiţi operatorii de extindere, şi anume: **C-restricţie, C-proiecţie, C-joncţiune, liniarizare şi grupare.**

Operatorii anteriori sînt introduşi prin intermediul unor exemple sugestive. În finalul lucrării sînt abordate modelele de stocare pentru obiecte complexe, precum şi aspecte legate de identitatea unui obiect în modelul relaţional.

Cuvinte cheie: baze de date relaţionale, relaţii structurate, forme normale, algebra relaţională, identitatea unui obiect.

INTRODUCERE

Tehnologia Bazelor de Date relaţionale şi-a demonstrat capacitatea de a implementa aplicaţii economice standard, cum ar fi gestiune de materiale, evidenţa contabilă, calcul salarii etc. Acest succes este datorat simplităţii matematice a modelului relaţional, bazat pe teoria mulţimilor. Cu toate acestea, există aplicaţii nestandard, care necesită funcţii de gestiune a datelor, cum ar fi bazele de cunoştinţe, birotică, proiectarea asistată de calculator etc., care nu pot fi abordate în cadrul modelului de date relaţional. O cerinţă comună a acestor aplicaţii este de a putea lucra eficient cu **obiecte complexe**. Aceasta este una din caracteristicile fundamentale ale orientării pe obiecte. Cu toate că modelul relaţional conţine concepte puternice, cum ar fi orientarea către mulţimi, este considerat insuficient pentru a suporta orientarea către obiecte, precum şi alte caracteristici (ca recursivitatea). Există două abordări pentru a rezolva problema **orientării pe obiecte**. Prima are în vedere înlocuirea modelului relaţional cu un model de date avînd o semantică mai bogată. Printre modelele de date propuse se pot enumera **modelul entitate-relaţie** [7], modelele de date semantice [16,10]. Unele dintre modelele propuse se bazează pe paradigme puternice, cum ar fi: **modelele de date bazate pe logică** [11], **funcţionale** [15] sau **orientate pe obiecte** [12]. Aceste modele nu au simplitatea modelului relaţional, iar cercetarea în scopul definirii unui model de date universal pentru toate tipurile de aplicaţii este de stringentă actualitate.

A doua abordare constă în extinderea modelului relaţional cu construcţii simple şi puternice. O extensie fundamentală este relaxarea restricţiei ca toate relaţiile să fie în 1NF (forma normală 1). Marele avantaj al acestei abordări este simplitatea şi, deci, viabilitatea ei. Totuşi, această dihotomie este simplistă întrucît unele dintre extensiile propuse pot fi, atît de semnificative, încît să ducă la definirea unui model de date complex, complet diferit.

În lucrare vor fi prezentate exemple de modele din ambele abordări.

1.O privire intuitivă asupra modelului de date orientat pe obiecte complexe

În acest capitol vor fi studiate următoarele probleme:

- măsura în care actualele SGBD-uri relaţionale pot manipula obiecte complexe;
- extensii ale algebrei relaţionale pentru prelucrarea eficientă a obiectelor complexe

1.1. Obiecte complexe în SGBD-urile relaţionale actuale

a. Obiecte atomice

Un **obiect atomic** este o unitate de date a unui tip particular, care furnizează o abstractizare către un nivel mai înalt de abstractizare din cadrul unui sistem de prelucrare de date. Structura obiectului atomic este ascunsă nivelelor superioare. De exemplu, un număr real este un obiect atomic. Alte exemple de obiecte atomice pot fi: documente, grafice şi imagini.

Obiectele atomice pot fi prelucrate prin intermediul unor operaţii specifice care "cunosc" structura acestora. Exemple de astfel de operaţii sînt: calculul numărului de cuvinte dintr-un document sau suprapunerea a două imagini. Cu toate că, definiţia unui domeniu în cadrul modelului relaţional nu este restrictivă, SGBD-urile standard oferă puţine tipuri de date predefinite ca **întreg, real, şir**, suficiente pentru cele mai multe dintre aplicaţiile economice. Aceste SGBD-uri pot prelucra obiectele atomice mari considerîndu-le şiruri de caractere de lungime mare. Astfel, SGBD-urile relaţionale pot stoca sau regăsi obiecte, dar interpretarea acestora este realizată de programe universale, care interfaţează cu SGBD-ul prin intermediul unui procesor. Limbajele de programare de nivel inalt sînt utilizate pentru a suplimenta SGBD-ul cu un sistem de tipuri mai bogat. Rezultă că, multe din funcţiile de prelucrare a datelor primitive sînt prelucrate de programele aplicative, şi nu de SGBD cum ar fi normal. Există o serie de probleme ridicate de această abordare, şi anume: codul pentru interpretarea obiectelor atomice poate fi stocat redundant în diferite aplicaţii; apoi, există o dependenţă puternică între obiect şi program; schimbarea formatului obiectului implică modificarea tuturor aplicaţiilor care

interpretează obiectul; în al treilea rînd, abordarea de mai sus este ineficientă deoarece programele aplicative, scrise în limbaje universale, au un control foarte mic asupra sistemului de operare, care manipulează obiectele. Dacă obiectul este mare, numai unele segmente din el vor fi stocate în spațiul de adresare a programului. Accesarea altor segmente va genera erori de paginare și posibile accese la disk.
O soluție ar putea consta în îmbogățirea sistemului de baze de date relaționale cu tipuri de date mai bogate. Astfel, SGBD-ul va putea interpreta structurile obiectelor atomice și va utiliza tehnici specifice pentru a le prelucra (indexare, buffering). Totuși, ar putea fi necesar un mare număr de tipuri primitive de date. Această soluție este, evident, nerealistă. În loc de a avea un număr fix de tipuri de date, o soluție mai bună este de a face ca tipurile SGBD-ului să fie extensibile conform cerințelor utilizatorului.

b. Obiecte complexe

Un **obiect complex** este un obiect ale cărui atribute pot fi ele însele obiecte. Structura unui obiect complex nu este liniară ca cea a unui tuplu relațional, ci este **ierarhică**. Astfel, un obiect complex poate fi intuitiv definit ca un obiect radacina și o ierarhie de subobiecte. Exemple de obiecte complexe pot fi: termeni complecsi (functori) din programarea logică [17], obiecte de birotică [1,4], obiecte de proiectare PAC [6]. De exemplu, în cazul unui sistem de birotică, se poate defini obiectul **dulap** care este un obiect ierarhic dotat cu un set de **sertare**, fiecare sertar conținînd un set de **dosare** și fiecare dosar avînd un număr de **documente**. Aceste obiecte pot fi ușor reprezentate în modelul relațional prin gruparea obiectelor de același tip și prin legarea obiectelor de tipuri diferite prin intermediul joncțiunilor. Pentru a modela un dulap sînt necesare patru relații, și anume:

Dulap (Dul#, Proprietăți...)
Sertar (Ser#, Dul#, Proprietăți...)
Dosar (Dos#, Ser#, Proprietăți...)
Document (Doc#, Dos#, Proprietăți...)

Problema fundamentală a acestei abordări este performanța scăzută a căutării întregului obiect complex. De exemplu, materializarea unui dulap particular va necesita trei joncțiuni ale relațiilor de mai sus. Problema poate fi redusă la optimizarea joncțiunilor pe chei străine (foreign keys). De exemplu, regăsirea eficientă a unui dulap și a tuturor sertarelor sale poate fi considerată ca un caz particular de optimizare a joncțiunii dintre relațiile Dulap și Sertar pe cheia străină Dul#. O cale pentru a realiza acest lucru este de a stoca tuplul unui dulap și Sertarele corespunzătoare împreună. Cu toate acestea, manipularea eficientă a obiectelor complexe este greu de realizat datorită existenței multiplelor feluri de căutare a unui obiect și ale componentelor sale. De exemplu, pot exista cereri de regăsire a unui Sertar pentru un Dulap particular și, concomitent, cereri de regăsire a tuturor Sertarelor create la o anumită dată. Astfel, orice organizare de tipul de mai sus va favoriza un tip de cereri, în schimb va defavoriza altele.
Extensiile SGBD-urilor relaționale, pentru a gestiona obiecte complexe, pun două probleme majore. În primul rînd, un model de date orientat pe obiecte complexe (OOC) trebuie să fie definit ca o extensie a modelului relațional. Extensia fundamentală este relaxarea restricției 1NF și permisiunea ca atributele să poată fi set_valuate sau tuplu_valuate (i.e., să poată avea valori tupluri sau mulțimi). Specificarea unui asemenea model include o algebră bine definită, de prelucrare a obiectelor complexe. În al doilea rînd, obiectele trebuie eficient implementate pentru a putea facilita anumite sabloane de căutare. În continuare, va fi studiată o extensie posibilă a modelului relațional definindu-se o algebră relațională extinsă.

1.2. Modelul de date orientat pe obiecte

Restricția 1NF asupra modelului relațional cere ca, obiectele complexe ierarhizate să fie normalizate (liniarizate). Relaxînd această restricție, obiectele complexe pot fi manipulate într-o manieră mult mai directă și, deci, eficientă. Obiectivul modelului orientat pe obiecte complexe (OOB) este de a combina avantajele modelelor relațional și ierarhic. Un astfel de model poate fi utilizat, atît la nivel conceptual, cît și la nivel intern. De asemenea, se pot defini forme normale pentru normalizare. De exemplu, [13, 14] descrie o formă normală pentru relații imbricate (relații ale căror atribute sînt relații), bazată pe dependențe multivaluate. Asemenea forme normale vor facilita proiectarea conceptuală a schemelor pentru obiectele complexe. În cele ce urmează, vom introduce într-un mod neformal noțiunea de obiect complex.

Definiție

Obiectele complexe sînt definite recursiv după cum urmează:

- întregi, reali, valorile logice și șirurile sînt obiecte atomice;
- dacă O1, O2,...,On sînt obiecte și dacă a1, a2,...,an sînt atribute distincte, atunci [a1:O1,...,an:On] este un obiect numit **obiect tuplu**;
- dacă O1, O2,...,On sînt obiecte, atunci [O1,O2,...,On] este un obiect numit obiect mulțime.

Tuplele pot conține atribute atomice, tuple sau set_valuate. Prima opțiune furnizează suport direct pentru relații normalizate. A doua opțiune permite utilizarea termenilor ierarhici [17]. În sfîrșit, a treia opțiune permite utilizarea relațiilor imbricate [2]. Exemple de obiecte sînt:

- un atom 10
- o mulțime {1,2,3}

- un tuplu [nume: Mircea, vîrsta:30]
- o relație {[nume: Mircea, vîrsta:30], [nume: Bebe, vîrsta:50]}
- un tuplu ierarhic [nume: [prim: Ion, ultim: Ionescu]]
- un tuplu cu relație imbricată [vin#: 210, disponibil: {[beci: 10, cant: 250], [beci: 20, cant: 500]}].

Definiția obiectelor complexe este generală. De exemplu, o mulțime de întregi este un obiect. Într-un SGBD, obiectele complexe, care au aceeași schemă, trebuie grupate în mulțimi, astfel încît operatorii de mulțimi pot fi aplicați eficient. Este evidentă necesitatea de a extinde conceptul de relație pentru ca atributele sale să aibă valori obiecte complexe.

Definiție

O **relație complexă** este o mulțime de tuple ale căror atribute pot avea ca valori obiecte complexe.

Abordarea implicată de definiția de mai sus permite ca, o algebră relațională complexă să poată fi definită prin extinderea algebrei relaționale obișnuite.

În continuare, vom defini o algebră relațională complexă prin intermediul următorului exemplu:

Exemplu

Baza de Date este **VINURI** și are două relații complexe: **VIN, VÎNZARE** avînd structurile următoare:

```
VINURI = [VIN : {[V#,
          Disponibil: {[Beci, Cant]},
          Podgorie,
          Recolta,
          Preț : {[Data, Mărime]},
VINZARE : {[Data :[Luna, An],
          Oraș,
          Client : {[Nume,
                    VIN : {[V#, Cant]}]}]}]
```

O reprezentare tabulară a obiectului VIN este:

V#	Disponibil		Podgoria	Recolta	Preț	
	Beci	Cant			An	Mărime
210	20	510	Panciu	1981	82	500.00
	35	600			83	400.00
					85	430.00
320	10	500	Odobești	1980	85	250.00
	20	450				

O algebră relațională complexă simplă cuprinde operatorii următori:

- operatori uzuali: reuniune, intersecție, produs cartezian;
- operatori complecși: C-restricție, C-proiecție, C-joncțiune;
- operatori de restructurare: liniarizare și grupare.

Definiție (C-restricție)

Este o operație care, aplicată asupra unei relații, produce o relație ale cărei tuple satisfac o condiție complexă.

Operatorul C – restricție este specificat astfel:

C-restricție (C-relație, C-condiție)

Condiția complexă este o expresie logică cu operatori AND, OR și NOT, aplicați predicatelor. Se pot distinge trei tipuri de predicate: simple, tuple imbricate, mulțimi imbricate. Predicatele simple sînt similare celor din algebra relațională standard. Predicatele cu tuple imbricate se aplică atributelor tuplelor imbricate. Operatorul "." este utilizat pentru a desemna un atribut într-un tuplu imbricat. De exemplu, Nume.Prim = "Ion".

Predicatele cu mulțimi imbricate se aplică atributelor tuplelor conținute în mulțimi imbricate. Pentru a desemna orice element dintr-o mulțime imbricată se va folosi operatorul binar "*", care are ca operand stîng un atribut set_valuat, iar ca operand - drept un nume de atribut. De exemplu, predicatul Preț*Mărime.

Exemple:

1. Se cer toate vînzările din 1986 din baza de date **VINURI** :

Operația este C-restrict (Vînzare, Data. An = "1986")

2. Se cer toate vinurile din Podgoria Panciu, cu un preț mai mic de 460.

Operația este:

C-restrict (Vin, Podgorie = "Panciu"
AND Preț*Mărime > 460)

Relația rezultată este:

V#	Disponibil		Podgoria	Recolta	Preț	
	Beci	Cant			An	Mărime
	20	510	Panciu	1981	82	500.00
	35	600				

Definiție (C-proiecție)

Operatorul produce o relație complexă dintr-o relație complexă de intrare, păstrînd numai atributele specificate și ștergînd duplicatele din fiecare obiect mulțime.

Specificarea operatorului este:

C-proiecție (C-relație, C-lista)

unde C-lista este o listă a atributelor care se păstrează. Recursiv, C-lista este o listă de A1,..., An, unde Ai este unul din numele-atribut [C-lista], și numele atribut {[C-lista]}, unde [] și {} desemnează un tuplu sau o structură de mulțime.

Exemple:

Operația C-proiecție (Vînzare, Data:[An], Oraș) produce o relație cu structura:

{[Data:[An], Oraș]}.

Operaţia C-proiecţie (Vin, Disponibil, Preţ: {[Mărime]}) produce rezultatul:

Disponibil		Podgoria	Recolta	
Beci	Cant			
20	510	Panciu	1981	500.00
35	600			400.00
				430.00
10	500	Odobeşti	1980	250.00
20	450			

Definiţia (liniarizare)

Operatorul normalizează o relaţie complexă.
De exemplu, operaţia:

Liniarizare (C–proiecţie (Vin, Disponibil, Podgorie))

va produce relaţia:

Beci	Cant	Podgoria
20	510	Panciu
35	600	Panciu
10	500	Odobeşti
20	450	Odobeşti

Definiţie (Grupare)

Operatorul grupează atributele unei relaţii în 1NF conform unei ierarhii date.
Operatorul Grupare este invers operatorului Liniarizare.
Operaţia este specificată prin:

Grupare(Relaţie, C-list)

unde C-list este similară cu cea de la C-proiecţie.
Totuşi, în C-list trebuie să apară toate atributele relaţiei. De exemplu, fie R o relaţie definită astfel:
R = Liniarizare(C-proiecţie(Vin, V#, Disponibil))
Operaţia de grupare: {[V#, Cant]}
Grupare (R, Beci, Vin: {[V#, Cant]})
grupează identificatorii de vinuri şi cantităţile respective pe Beciuri. Relaţia complexă rezultată este următoarea:

Beci	Vin	
	V#	Cant
10	320	500
20	210	510
	320	450
35	210	600

Definiţie (C-joncţiune)

Operaţia combină două relaţii complexe într-o a treia concatenînd fiecare pereche de tuple care satisfac o condiţie asupra atributelor de la nivelul întîi.
Operaţia de C-joncţiune se specifică astfel:

C-joncţiune(C-relaţie1, C-relaţie2, J-condiţie)
J-condiţie este o expresie logică, folosind AND, OR şi NOT, de predicate de forma A1 op A2, unde A1 şi A2 sînt nume de atribute din C-relaţie1 şi, respectiv, C-relaţie2, iar op este un operator de comparaţie corespunzător tipurilor atributelor A1 şi A2.

Limitarea principală a C-joncţiunii este că, atributele utilizate în condiţie trebuie să apară la primul nivel al relaţiei complexe. De exemplu, nu se pot joncţiona direct relaţiile VIN şi VINZARE pe atributul V#, întrucît acesta în VINZARE nu este la primul nivel. Deci, VINZARE trebuie întîi restructurată. În acest caz, C-joncţiunea este:

C-joncţiune(Vin, Liniarizare(Vînzare), V#=V#)

1.3. Tehnici de implementare pentru obiecte complexe

În acest capitol, vor fi prezentate două tehnici de implementare pentru obiecte complexe, suportate la nivel conceptual. Prima tehnică, denumită model de stocare directă, mapează obiectele direct în spaţiul de adrese fizice, astfel încît obiectele şi subobiectele sînt grupate împreună. A doua tehnică, denumită model de stocare normalizat, are cîteva variante. Ideea este de a descompune şi de a memora valorile atomice ale obiectelor de acelaşi tip în fişiere liniarizate şi de a captura conexiunile dintre obiecte şi subobiecte prin intermediul unor structuri numite **indecşi** de joncţiune. Pentru orice tehnică de implementare se va presupune ca fiecărui tuplu îi este asigurat un surogat de identitate, numit TID (tuplu identifier). Un TID este o valoare unică în cadrul unei relaţii care se creează de către sistem la instanţierea unui tuplu şi pe care sistemul nu o mai modifică. TID-urile permit actualizări eficiente şi reorganizări de fişiere întrucît referinţele nu implică pointeri fizici. Multe dintre SGDB-uri, incluzînd INGRES şi System R, suportă noţiunea de TID.

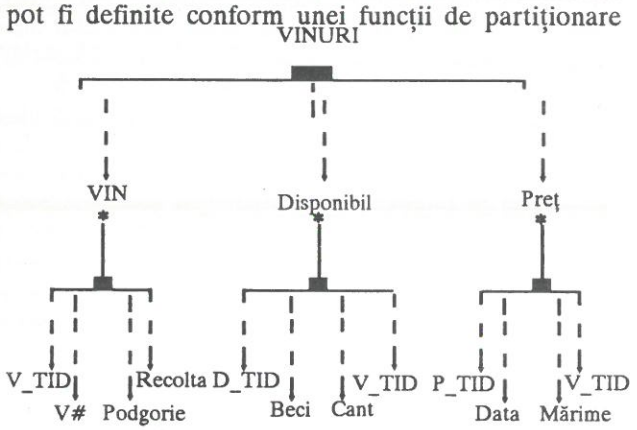
1.3.1. Modelul de stocare directă

Dacă o bază de date este compusă din obiecte de tip mulţime, modelul de stocare directă va memora fiecare obiect-mulţime (care poate fi un set imbricat) într-un fişier separat. Fiecare înregistrare din fişier reprezintă un obiect complex. Există cîteva soluţii de grupare a atributelor la un obiect complex, bazate pe o ierarhie a seturilor imbricate. O soluţie simplă şi consistentă este ordonarea în adîncime (depth_first ordering).
Gruparea înregistrărilor într-un fişier poate fi dată numai pe baza atributelor obiectelor de la rădăcină. Fişierul VIN poate fi grupat pe V#, Podgorie şi/sau Recolta, utilizînd o structură de fişier mono sau multiatribut. De aici, rezultă că accesul la obiecte bazate pe atribute diferite de cele ale obiectelor-rădăcină, trebuie realizat cu structuri auxiliare sau prin căutări secvenţiale.
Avantajul fundamental al acestei abordări este acela al

regăsirii eficiente a întregului obiect complex. Comparat cu maparea unei scheme relaționale în care fiecărei relații îi corepunde un fișier, modelul de stocare directă evită multe din joncțiuni. Alt avantaj puternic al modelului este că faza de compilare a cererilor pe obiecte complexe conceptuale este simplificată datorită corespondenței biunivoce între obiectul conceptual și obiectul intern.
Principalul dezavantaj al acestei abordări este că, regăsirea diferitelor subobiecte este ineficientă datorită faptului că acestea sînt grupate pe criterii topologice.

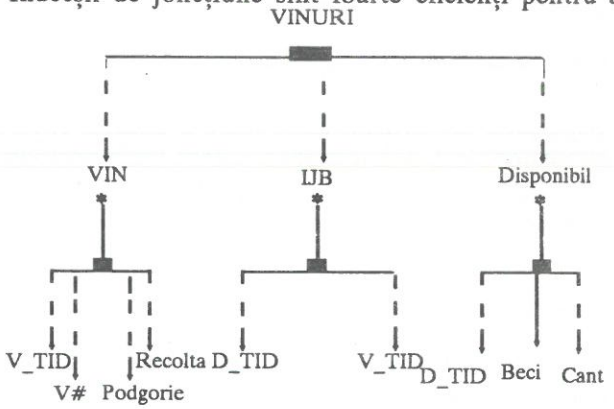
1.2.2. Modelul de stocare normalizat

În acest model, obiectele complexe sînt descompuse în relații normalizate 1NF. De exemplu, obiectul complex VIN poate fi descompus în trei relații liniare: VIN, Disponibil și Preț, TID-urile pentru acestea sînt notate V-TID, D_TID și respectiv, P_TID. Conexiunea dintre un tuplu VIN și subobiectele sale este menținută prin adăugarea TID-ului V_TID la relațiile asociate subobiectelor. Schema normalizată pentru obiectul complex VIN este:
Diferite variante ale modelului de stocare normalizat pot fi definite conform unei funcții de partiționare

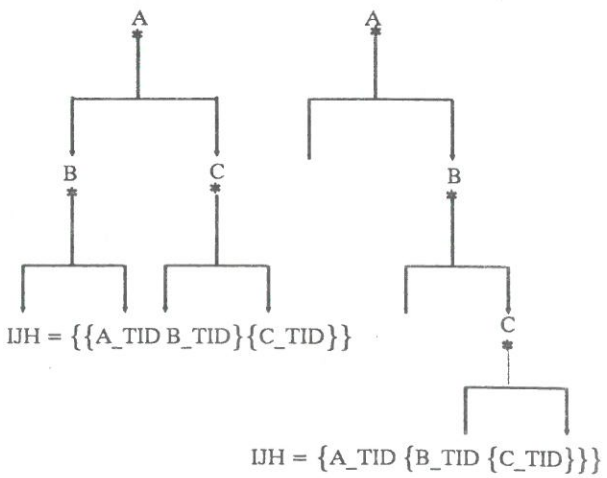


verticală. Aceasta descompune o relație într-unul sau mai multe fișiere, fiecare fișier conținînd un număr oarecare de atribute. Cele mai frecvente variante ale modelului grupează toate atributele unei relații în același fișier. Totuși, indiferent de varianta de stocare aleasă, compunerea de obiecte complexe necesită joncțiuni multiple, care sînt operații costisitoare. O cale de a realiza regăsiri, într-un mod mai eficient, este de a utiliza indecși de joncțiune care stochează într-un mod uniform și compact structurile obiectelor complexe. Există două versiuni de indecși de joncțiune: un index de joncțiune binar pentru obiecte simple și o versiune generalizată, numită index de joncțiune ierarhic.
Un index de joncțiune binar este o abstractizare a joncțiunii dintre două relații. El este o mulțime de perechi de TID-uri, cîte unul pentru fiecare relație, astfel încît, tuplele corespunzătoare se potrivesc predicatului de joncțiune.
Un index de joncțiune este implementat ca o relație binară. De exemplu, conexiunea dintre relațiile VIN și Disponibil, din figura anterioară, este realizată prin stocarea explicită a TID-ului pentru VIN în Disponibil. Această conexiune poate fi stocată explicit în indexul de joncțiune (D_TID, V_TID), cum se arată în figura următoare:

Indecșii de joncțiune sînt foarte eficienți pentru a



optimiza joncțiunile, în principal, datorită faptului că aceștia sînt separați de baza de date, iar dimensiunea lor este suficient de mică pentru a fi ținuți în memoria principală. Ei pot fi utilizați sistematic pentru capturarea joncțiunilor care materializează obiecte complexe, precum și pentru optimizarea altor feluri de joncțiuni.
Pentru a se permite manipularea obiectelor complexe, noțiunea de index de joncțiune binar a fost extinsă la **index de joncțiune ierarhic**, care poate captura structura unui obiect complex utilizînd TID-urile relațiilor conectate, care formează obiectul.
Un **index de joncțiune ierarhic** este un obiect complex în care fiecare obiect_ tuplu este înlocuit prin TID-ul său, și fiecare set de atomi este ignorat. Două exemple de asemenea indecși sînt:



De aceea, în loc de a utiliza cîțiva indecși de joncțiune binari este mai eficientă folosirea unui singur index de joncțiune ierarhic. Ca în modelul de stocare directă, un index de joncțiune ierarhic poate fi grupat numai de TID-ul rădăcină. Cînd un TID rădăcină al unui obiect complex este obținut, atunci întreaga structură a obiectului se poate obține direct. Indecșii de joncțiune ierarhici sînt mai eficienți decît indecși de joncțiune binari în regăsirea unui întreg obiect. De aceea, cele două tipuri de indecși trebuie folosite în maniera complementară.

1.4 Facilități de identificare a obiectelor

Utilizarea identificatorilor pentru obiecte-tuplu, numiți TID-uri, este esențială pentru implementarea obiectelor structurate ierarhic. Utilizarea identificatorilor de obiecte este, în general, limitată la nivelul intern al sistemului. TID-urile nu sînt vizibile la nivelurile conceptual și extern (cu excepția notabilă a lui INGRES). Noțiunea de **identitate a unui obiect** este proprietatea unui obiect de a se distinge de oricare alt obiect, fără a se ține seama de conținutul, localizarea sau adresa acestuia.
În continuare, se va aborda problema încorporării noțiunii de identitate a unui obiect la nivelul conceptual al unui SGBD.

1.4.1 Identitatea unui obiect în Modelul Relațional

Obiectele primitive din modelul relațional sînt relațiile, tuplele și atributele. Relațiile sînt identificate prin numele extern, definit de utilizator. Atributele sînt identificate prin valorile lor. O astfel de identitate pentru relații și atribute este trivială. Identitatea unui tuplu este mai relevantă. O soluție veche este de a utiliza unul sau mai multe atribute ale unui tuplu ca o cheie pentru relație. Această soluție se aplică, de asemenea, la modelul OOC, unde numai tuplele de nivelul întîi au nevoie de identificare.
Această abordare pune unele probleme datorită rolului dual jucat de atributele cheie ca identificatori și ca descriptori. Problema majoră a utilizatorului este că atributele cheie nu pot fi manipulate ca celelalte atribute. În particular, un atribut cheie nu poate fi actualizat la fel de simplu ca un atribut necheie. De exemplu, un nume de țară poate fi utilizat ca o cheie pentru relația ȚĂRI. Numele de țară poate fi utilizat ca o cheie străină în relația ORAȘE. Totuși, în cazul unei revoluții numele unei țări poate fi schimbat. Toate tuplele corespunzătoare orașelor din țara respectivă se vor actualiza. Altă problemă care apare frecvent este cea a inexistenței unor atribute care să identifice unic toate tuplele. De exemplu, într-o bază de personal pot exista două persoane cu aceleași nume și prenume. În acest caz, se poate introduce un atribut artificial, atributul numărul persoanei. Astfel de informații artificiale sînt greu de generat, necesitînd uneori informații adiționale despre obiect. O soluție a acestei probleme este facilitatea implicită de identitate în cadrul modelului de date.

1.4.2 Identitatea unui obiect în modelul OOC

Un obiect O este un cuplu (identificator, valoare) în care:
- identificatorul furnizează o referință unică și invariantă;
- valoarea lui O depinde de tipul său și este una din următoarele:
  1. dacă tipul este atom, valoarea este un element al unui sistem sau provine dintr-un domeniu de valori specificat de utilizator;
  2. dacă tipul este tuplu, valoarea este [a1:I1,...,an:In], unde fiecare ai este un atribut distinct, iar Ii este un identificator;
  3. dacă tipul este set, valoarea este {I1, I2,...,In}, unde Ii sînt identificatori distincți.

Această definiție permite exprimarea obiectelor graf-structurate. Orice obiect poate avea mai mulți părinți care se referă la el prin identificatorul acestuia. În comparație cu modelul ierarhic, obiectele partajate nu sînt replicate. Presupunînd că baza de date este o mulțime de relații complexe, pot apărea două feluri de partajare: partajarea subobiectelor în cadrul unei aceleeași relații complexe și coreferențierea obiectelor, dintr-o relație complexă, în altă relație complexă.
Încorporarea identității, ca o parte integrantă a unui model orientat pe obiecte complexe, necesită extensii ale limbajului de manipulare a obiectelor cu identitate și tehnici de implementare adiționale pentru a putea prelucra obiectele partajate. Principala extensie a limbajului trebuie să asigure diferite căi pentru a testa egalitatea a două obiecte. În [11] sînt propuse trei tipuri de egalitate: de identitate, superficială și de adîncime. Astfel:
1. două obiecte sînt egale prin **identitate**, dacă au același identificator;
2. două obiecte sînt egale superficial, dacă au valori egale;
3. sînt egale în adîncime două obiecte dacă:
   (atomice): aceeași valoare;
   (mulțime): elementele lor sînt egale în adîncime două cîte două;
   (tuplu): valorile luate de aceleași atribute sînt egale în adîncime.

Această definiție este recursivă și permite compararea subobiectelor corespunzătoare a două obiecte.
Similar egalității, se pot defini două tipuri de operatori de copiere:
- copierea superficială, care duplică numai primul nivel al unui obiect;
- copierea de adîncime, care duplică obiectul împreună cu subobiectele sale.

În aceeași manieră se pot defini operatorii orientați pe mulțime. De exemplu, se pot defini două tipuri de

operatori de reuniune: bazați pe valoare sau bazați pe identitate. Limbajele rezultate se depărtează semnificativ de la tipul de limbaje relaționale, care sînt tradițional bazate pe valoare, și nu pe identitate.

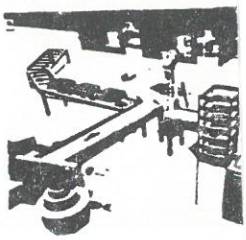
BIBLIOGRAFIE

1. ADIBA, M., NGYUEN, G.T.: An Algebra for Non Normalized Relation. În: ACM Int. Symp. on PODS, 1984, pp.45-52.
2. BANCILHON, F., RICHARD, P.: On Line Processing of Compacted Relations. În: Int. Cinf. on VLDB, 1982, Mexico, pp.202-208.
3. BANCILHON, F., ș.a.: FAD, a Powerful and Simple Database Language. În: Proc. 13th VLDB, e. Brighton, 1987, pp.149-160.
4. BANERJEE, J., ș.a.: Data Model Issues for Object Oriented Applications. În: ACM TODS, Vol.5, Ianuarie, 1987, pp.63-69.
5. BANERJEE, J., ș.a.: Semantics and Implementation of Schema Evolution in Object - oriented Databases. În: ACM SIGMOD, 1987, San Francisco, pp.190-202.
6. BATORY, D.S., KIM, W.: Modelling Concepts for VLSI CAD Objects. În: ACM TODS, Vol.1, 1976, pp.75-93.
7. CHEN, P.P.: The Entity - Relationship Model: Towards a Unified View of Data. În: ACM TODS, Vol.1, martie, 1976, pp.103-110.
8. CHEN, Q., GARDARIN, G.: An Implementation Model for Reasoning with Complex Object. În: INRIA, nr.803, Franța, februarie 1988, (Raport de Cercetare).
9. FISHMAN, D.H., BEECH, D. ș.a.: IRIS: An Object - Oriented Database System. În: ACM Transactions on Office Information System, nr.1, ianuarie, 1987, pp.25-33.
10. KING, R., MCLEOD, D.: Semantic database Models. În: Database Design, Editor S.B. Yao, 1985, pp.120-141.
11. KHOSHAFIAN, S.: Object Identity, Proc. of OOPSLA 1986, Conference Portland, OR, octombrie, 1986 (XC).
12. KUPER, G.M., VARDI, M.Y.: On the Expressive Power of the Logic Data Model. În: ACM SIGMOD, 1985, Austin, Texas, pp.73-81.
13. MAIER, D., STEIN, J., OTTIS, A.: Development of an Object - Oriented DBMS. În: Int. Conf. on OOPSLA, 1986, Portland, pp.132-143.
14. OSZOYGLU, Z.M., YUAN, L.-Y.: A Normal Form for Nested Relation. În: ACM PODS, 1985, Portland, Oregon, pp.190-200.
15. OSZOYOGLU, G., OSZOYOGLU, Z.M.: A Language and a Physical Organization Technique for Summary Tables. În: ACM SIGMOD, 1987, Austin, Texas, pp.481-493.
16. SHIPMAN, D.: The Functional data Model and the Data Language DAPLEX. În: ACM TODS, Vol.6, 1981, pp.302-317.
17. TSUR, S., ZANIOLO, C.: On the Implementation of GEM: Supporting a Semantic Data Model on a Relational Back-end. În: ACM SIGMOD, Boston, 1984, pp.233-245.
18. ZANIOLO, C.: The Representation and Deductive Retrival of Complex Objects. În: Proc. VLDB, Stockholm, Vol.11, 1985, pp.302-321.

New Jersey Institute of Technology

CALL FOR PAPERS

9TH INTERNATIONAL CONFERENCE ON CAD/CAM, ROBOTICS AND FACTORIES OF THE FUTURE



cars & fo

18-20 August 1993
New Jersey

Sponsored by
International Society for Productivity Enhancement (ISPE), USA

OBJECTIVES: The main objective of this conference is to bring together researchers and practicing engineers from gc ernment, industries and academia in the multi-disciplinary aspects of advanced manufacturing systems utilizing creati engineering, design, artificial intelligence, robotics, database management, prototyping and computer integrated manuf turing.

SUBJECTS: Technical areas for the conference include (but are not limited to):

Computer-Aided-Engineering: CAD, FEM, Process Planning, NC Programming, QC Planning, Simulation, Analys Computer Graphics.

Computer-Aided-Manufacturing: FMS, Integration of CNC, Machine Vision, Controls, Materials, Data Manageme Automated Packing, Material Handling.

Concurrent Engineering: Design for Manufacturability, Design Theory, Optimization, Value Engineering, Cos Productivity, Design Axioms, Reliability and Safety.

Knowledge Automation: Artificial Intelligence, Expert Systems.

Factories of the Future: Intelligent Warehouse Systems, Cellular Systems, Production Planning & Control, Plant Desi Productivity Enhancement, Organization, Socioeconomic Issues, Employee Training and Environmental Issues..

Robotics and Manipulators: Mechanical Design, Control, Trajectory Planning, End Effectors, Sensory Devices, Mobili Applications, Work Cells, Standardization.

Emerging Technological Processes: Manufacturing of Ultra Precision Components, Processing of Advanced Materia Space Technologies, Non-Traditional Machining and other Miscellaneous Topics.

Manuscripts of full length papers accepted and presented at the conference will be published in a bound book form.

Selected papers from the conference will be published in a special issue of the International Journal of Syster Automation: Research and Applications (SARA), the official journal of the International Society for Productiv Enhancement (ISPE).

Paper Submissions: Authors are invited to submit three copies of a minimum of 500 words abstract(s) on the above su jects. An abstract should contain the title of the paper, full name(s) and addresses of all authors. Deadline for submissi of abstract: November 30, 1992. Send abstract to:

Dr. Raj S. Sodhi, Director
Manufacturing Engineering Programs
New Jersey Institute of Technology
Newark, New Jersey 07102, U.S.A
Telephone: (201) 596-3185
Fax#: (201) 596-6056
E-Mail: SODHI @ADMIN1.NJIT.EDU