

RDBMS vs. OODBMS

Prof.dr. Gheorghe SABĂU
Catedra de Informatică Economică, A.S.E. București

The author uses a comparative presentation of Relational Database Management Systems (RDMS) and Object Oriental Database Management Systems (OODBMS). The presentation of the two types of systems was made having in view: The evaluation rules of the two types of systems elaborated by E.F. Codd and OMG; Data Modeling; The objectives of the Systems. In the end the article contains conclusions referring to the two types of systems.

Introducere

Prezentarea asemănărilor și deosibirilor dintre modelarea relațională și modelarea orientată obiect a datelor este semnificativă și de mare importanță atât pentru proiectanți de baze de date cât și pentru utilizatori.

Proiectanții, cunoscând foarte bine modelul relațional și apoi sesizând asemănările și deosebiriile dintre cele două moduri de abordare a modelării datelor, vor putea valorifica și folosi experiența dobândită anterior ca bază substanțială pentru înțelegerea și însușirea metodologiei de proiectare a bazelor de date orientate obiect. Totodată, prin cunoașterea asemănărilor și deosebirilor dintre cele două moduri de abordare apare posibilitatea conversiei unui model relațional în obiectual și invers. De altfel, o astfel de practică a regăsirii în mod frecvent.

În cele ce urmează vom recurge la o tratare comparativă SGBDR și SGBDOO luând în considerare regulile de evaluare a celor două tipuri de sisteme precum și modelarea datelor sub aspectul conceptelor folosite și obiectivelor urmărite.

A. Regulile de evaluare a unui SGBDR

Este știut faptul că în prezent există o multitudine de SGBD relaționale, ele fiind chiar de ordinul sutelor. Unele dintre acestea sunt declarate de realizatori ca fiind relaționate însă, în realitate ele nu întrunesc criteriile de a putea fi considerate cu adevărat relaționale.

E.F. Codd, din dorința de a menține nealterat modelul relațional, a elaborat 13 reguli după care ar putea fi apreciat orice SGBD dacă este cu adevărat sau nu relațional [3, 4]. În cele ce urmează, din considerente de lipsă de spa-

țiu, vom recurge mai mult la o simplă enumerare a acestora.

Regula O – Regula fundamentală

Un SGBD pentru a fi considerat relațional el trebuie să fie capabil să gestioneze în întregime bazele de date prin capacitățile sale relaționale.

Regula amintită accentuează un aspect esențial și anume că un SGBD nu trebuie să recurgă la nici un fel de operații nerelaționale pentru a realiza sarcinile ce îi revin cu privire la definirea, utilizarea și manipularea datelor.

Regula 1 – Regula de nonsubversiune

Dacă un sistem relațional are un limbaj de nivel scăzut, acel nivel scăzut nu poate fi utilizat pentru a submina sau ocoli regulile de integritate și constrângerile exprimate în limbajul relațional de nivel mai înalt.

O astfel de regulă imprimă constrângeri în sensul că întregul acces la baza de date trebuie să fie controlat de către SGBD, astfel încât integritatea bazei de date să nu poată fi compromisă fără cunoștința utilizatorilor sau administratorului bazei de date.

Regula 2 – Reprezentarea informațiilor

La nivel logic, toate informațiile dintr-o bază de date relațională sunt reprezentate într-un singur mod și anume prin valorile din tabele.

O astfel de regulă impune ca toate informațiile, chiar și meta-datele, conținute în dicționarul de sistem să fie stocate ca relații și gestionate de către aceleași funcții operaționale care ar fi utilizate pentru întreținerea datelor.

Regula 3 – Reactualizarea vederilor

Toate vederile care sunt teoretic reactualizate pot fi reactualizate și de către sistem.

Prin această regulă se stabilește că, dacă o vedere poate fi teoretic reactualizată, atunci

sistemul de gestiune trebuie să fie capabil de a efectua reactualizarea respectivă.

De remarcat faptul că deși Codd precizează o astfel de regulă, practic nici un sistem nu acceptă cu adevărat această caracteristică, deoarece nu au fost încă descoperite condițiile pentru identificarea tuturor vederilor care pot fi teoretic reactualizate.

Regula 4 – Tratarea sistematică a valorilor null

Valorile null, deosebite de șirul nul de caractere sau de un șir de caractere nule, ca și de zero sau orice alt număr, sunt acceptate pentru a reprezenta informațiile lipsă și cele care nu pot fi aplicate în mod sistematic, indiferent de tipul de date.

Regula 5 – Independența de integritate

Constrângerile de integritate specifice unei anumite baze de date relaționale trebuie să poată fi definite în sublimbajul relațional de date și stocate în catalog, sau în programele de aplicație.

Regula 6 – Accesul garantat

Se garantează că fiecare element de dată/valoare atomică dintr-o bază de date relațională este accesibil din punct de vedere logic, prin apelarea la o combinație de nume de tabel, valoare a cheii primare și nume de coloană.

Regula 7 – Catalog dinamic on-line, bazat pe modelul relațional

Descrierea bazei de date este reprezentată la nivel logic în același mod ca și datele obișnuite, astfel încât utilizatorii autorizați pot aplica la interogarea acestora același limbaj relațional ca cel aplicat datelor curente.

Regula 8 – Sublimbaje de date cuprinzătoare

Un sistem relațional poate accepta mai multe limbaje și diverse moduri de utilizare a terminalelor. Totuși, trebuie să existe cel puțin un limbaj ale cărui instrucțiuni să poată exprima următoarele chestiuni:

(1) definirea datelor, (2) definirea vederilor, (3) manipularea datelor, (4) constrângerile de integritate, (5) autorizarea, (6) limitele tranzacțiilor (begin, commit și rollback).

Regula 9 – Operații de inserare, reactualizare și ștergere de nivel înalt

Capacitatea de tratare a unei relații de bază

sau a unei relații derivate (adică o vedere) ca pe un singur operand se aplică nu numai reștergerii de date, ci și inserării, reactualizării și ștergerii acestora.

Regula 10 – Independența fizică de date

Programele de aplicație și activitățile de la terminal rămân logic intacte, ori de câte ori sunt făcute modificări, fie în reprezentările de stocare, fie în metodele de acces.

Regula 11 – Independența logică de date

Programele de aplicație și activitățile de la terminal rămân logic intacte, ori de câte ori sunt efectuate în tabelele de bază orice fel de modificări privind păstrarea informațiilor, care, teoretic, permit deteriorarea acestora.

Regula 12 – Independența de distribuție

Sublimbajul de manipulare a datelor dintr-un SGBD relațional trebuie să permită programelor de aplicație și înregistrărilor să rămână aceleași din punct de vedere logic, dacă și ori de câte ori datele sunt centralizate sau distribuite fizic.

B. Reguli de evaluare a unui SGBDOO

Așa după cum Codd E.F. [3,4] a definit o serie de reguli pentru ca un SGBD să fie cu adevărat relațional, tot la fel un grup de lucru să fie de mare influență a publicat, sub denumirea de „Manifestul sistemelor de baze de date orientate pe obiecte”, o serie de cerințe sau principii pe care trebuie să le îndeplinească un SGBD pentru a putea fi considerat obiectul [3]. În concordanță cu Manifestul sistemelor de baze de date orientate pe obiecte, sunt definite 13 principii ale AGBDOO, dintre care unele sunt obligatorii iar altele opționale.

1. Posibilitatea definirii structurilor complexe.

O astfel de cerință presupune capacitatea de a defini structuri complexe plecând de la tipuri de date atomice folosind o serie de tipuri de constructori, iar aceștia să fie ortogonali, în sensul că fiecare constructor trebuie să se aplice oricărui obiect. Exemplu, dacă folosim constructori de forma SET (TUPLE ()), LIST (TUPLE ()) atunci trebuie să avem posibilitatea de a folosi și formele TUPLE (SET ()) și TUPLE (LIST ()).

Dintre cei mai folosiți constructori amintim:

- ROW (n_1t_1, \dots, n_nt_n), unde un tip reprezintă un rând sau tupleu de n câmpuri au câmpurile n_1, \dots, n_n de tipurile t_1, \dots, t_n ;

- ARRAY: Tipurile array suportă o metodă „array index” pentru a permite utilizatorilor să acceseze articolele array (colecției);

- seturi și multiseturi (Sets and multisets).

Obiectele set pot fi comparate utilizând setul de metode tradițional $<, \leq, =, \geq, >$.

Totodată, două obiecte set (având elementele de același tip) pot fi combinate pentru a forma un nou obiect folosind operatorii \cup, \cap și $-$ (diferență).

- Liste: Operațiile de liste tradiționale include capul de listă (head) care returnează primul element, coada listei (tail) care returnează adresa ultimului element din listă, inserare sau adăugare de noi elemente în listă și respectiv excluderea unui element din listă. Mai există o serie de operatori, cum ar fi operatorii agregați (COUNT, SUM, AVG, MAX, MIN) și operatori pentru conversia de tipuri (exemplu, pentru conversia unui obiect multiset într-un obiect set prin eliminarea duplicatelor).

2. Identitatea obiectului, adică SGBD trebuie să ofere posibilitatea identificării, fără ambiguitate, a oricărui obiect din baza de date. În acest sens cu ocazia încărcării oricărui obiect în baza de date se va recurge la generarea unui identificator unic al obiectului, numit OID.

3. Încapsularea, presupune faptul că SGBD trebuie să dispună de capacitatea de a încapsula datele și metodele aparținând obiectelor iar utilizatorii pot avea acces la ele doar printr-o interfață ce citează o anumită metodă. La rândul său atât metodele cât și datele pot fi definite publice (+), protejate (\neq) sau private (-).

4. Tipuri și/sau clase. Cele două concepte trebuie să fie ambele prezente. Primul concept reprezintă un mecanism de verificare pentru acuratețea programelor în timpul compilării. Al doilea concept reprezintă un mecanism care colectează extensiile de obiecte și le definește implementarea lor. Invers, nu e necesar să fie două forme diferite de a exprima tipuri și clase și astfel e posibil să exprimăm un concept în contextul celuilalt.

5. Clase și/sau tipuri de ierarhii, în contextul moștenirii, evidențiază faptul că un SGBD trebuie să asigure ca un subtip sau subclasă să poată moșteni atributele și metodele de la superclasele sau supertipurile lor.

6. SGBD trebuie să ofere suport pentru legarea dinamică, în sensul că metodele trebuie să se aplice la obiecte de diferite tipuri iar implementarea unei metode va depinde de tipul de obiect căruia i se aplică. Acest aspect presupune faptul că sistemul nu poate lega denumirea metodelor până în momentul execuției (legea dinamică).

7. SGBD trebuie să ofere un limbaj de manipulare a datelor LMD cu completitudine de calcul. În acest sens, de exemplu, se apreciază că standardul SQL3 posedă completitudine de calcul.

8. Extensibilitatea mulțimii tipurilor de date, ceea ce presupune capacitatea de a defini noi tipuri bazate pe cerințele utilizatorilor.

9. Durabilitatea sau persistența datelor, ceea ce presupune capacitatea sistemului de a asigura / suporta persistența datelor. La fel precum în situația SGBD convenționale, datele trebuie să rămână după finalizarea aplicației care le-a creat. Deci, utilizatorul nu trebuie să deplaseze sau să copieze explicit datele, pentru a le putea refolosi.

10. SGBD trebuie să fie capabil și să ofere facilități în ceea ce privește operarea și gestionarea unor volume mari de date (baze de date de mari dimensiuni).

11. Asigurarea accesului concurent la aceleași resurse.

12. Asigurarea capacității de refacere în cazul unor căderi fizice de hardware sau software, asemănător sistemelor convenționale.

13. Asigurarea unor limbaje de acces de cereri de nivel înalt cu multiple facilități de interogare a bazei de date.

Manifestul bazelor de date orientate direct mai face referiri la câteva caracteristici opționale, care sunt interesante și folositoare dar nu esențiale pentru un SGBDOO, dintre acestea enumerăm: moștenirea multiplă, distribuția datelor într-o rețea, posibilitatea verificării unui program în timpul compilării, managementul tranzacțiilor și mecanisme pentru ma-

nagementul versiunilor.

C. Comparații între abordarea obiectuală și cea relațională privind modelarea datelor

Deosebirea esențială între cele două tipuri de modelare a datelor o reprezintă încapsularea

în obiect atât a stării cât și a comportamentului acestuia, în vreme ce modelul Entitate – Relație evidențiază numai starea și nimic despre comportament.

În tabelul 1 se prezintă o comparație între principalele concepte utilizate în modelarea obiectuală și relațională a datelor.

Tabelul 1. Compararea BDOO și BDR sub aspectul conceptelor folosite

Modelul orientat pe obiecte	Modelul relațional	Diferențe
Obiect	Entitate	Obiectul precizează și comportamentul
Clasa de obiecte	Tipuri de entități	Clasa de obiecte include și comportamentul comun obiectelor din clasa respectivă
Ierarhia de clase	Schema bazei de date	Ierarhia de clase implică moștenirea iar schema implică chei externe
Instanță de clasă	Entitate, triplu sau înregistrare	Instanța poate avea un caracter mai restrictiv
Atribut	Atribut	Fără diferențe
Relații	Relații	Fără diferențe Au semnificația de descrieri, însă în BDOO moștenirea include atât starea cât și comportamentul
Mesaje / interfață	Nu există	
Încapsulare	Nu există	
Identificatorul de obiect (OID)	Cheie primară	În modelul relațional dacă nu se definește cheia primară sistemul generează în mod automat un identificator

Deosebirile dintre modelul BDOO și modelul BDR pot fi evidențiate și sub aspectul obiec-

tivelor urmărite sau prin alte caracteristici, așa cum se poate observa din tabelul 2.

Tabelul 2. Compararea BDOO și BDR sub aspectul obiectivelor urmărite

BDOO	BDR
• Obiective principale: încapsularea și independența datelor.	• Obiectiv principal: asigurarea independenței datelor față de programele de aplicații.
• Independența claselor: pot fi reorganizate fără a afecta modul de folosire a lor.	• Independența datelor: datele pot fi reorganizate și modificate fizic fără a afecta modul de folosire.
• BDOO stochează date și metode.	• BDR stochează numai date.
• Încapsularea: datele pot fi folosite numai prin metodele claselor.	• Partiționarea datelor: datele pot fi partiționate în funcție de cerințele și specificul aplicațiilor utilizatorilor.
• Obiecte active: obiectele sunt active. Cerințele cauzează obiectelor executarea metodelor acestora.	• Date pasive: datele sunt pasive. Anumite operații limitate pot fi în mod automat antrenate când datele sunt folosite.
• Complexitate: structura datelor poate fi complexă, implicând multiple tipuri de date.	• Simplitate: utilizatorii percep datele sub formă de coloane, rânduri / tupluri și tabele.
• Date înlănțuite. Datele pot fi înlănțuite așa încât metodele claselor oferă performanțe sporite. Datele structurate precum și BLOB (binary large objects) sunt folosite pentru sunet, imagine, video etc.	• Tabele separate: fiecare relație / tabelă este separată. Operatorul de JOIN referă date din tabele separate.
• Neredundanța metodelor: neredundanța datelor și metodelor sunt realizate prin încapsulare și moștenire. Moștenirea ajută la reducerea redundanței metodelor.	• Neredundanța datelor: Normalizarea datelor are ca scop de a elimina sau reduce redundanța datelor. Ea este folosită în faza de proiectare a bazei de date și nu în faza de dezvoltare a aplicațiilor.
• Optimizarea claselor: datele pentru un obiect	• Performanța BDR este legată de nivelul de complexi-

BDOO	BDR
pot fi interlegate și stocate împreună, astfel încât ele pot fi accesate împreună de mecanismul de acces.	tate a structurii datelor.
<ul style="list-style-type: none"> • Modul conceptual consistent: modelele folosite pentru analiză, proiectare, programare și accesul bazei de date sunt similare. Conceptele aplicațiilor sunt în mod direct reprezentate prin clasele de obiecte. 	<ul style="list-style-type: none"> • Model conceptual diferit: modelul structurii datelor și acces reprezentat prin tabele și JOIN-uri este diferit de cel în analiză, proiectare și programare. Proiectul trebuie să fie convertit în tabele relaționale și acces conform SQL.

Concluzii

Din literatura de specialitate precum și din cele prezentate pot fi desprinse o serie de concluzii cu privire la SGBDR și SGBDOO, dintre care amintim:

- Bazele de date relaționale au ca obiectiv principal asigurarea independenței datelor. Datele normalizate sunt separate de prelucrări iar prelucrările corespunzătoare satisfacerii cerințelor informaționale nu este obligatoriu să fie în totalitate predefinite, deci se acceptă și cerințele ad-hoc;
- Bazele de date orientate direct au ca obiectiv principal încapsularea, fiind stocate împreună datele și metodele. Ele sunt inseparabile. Se spune că avem de a face cu independența claselor nu cu independența datelor;
- Nevoia unui SGBDOO și nu unul relațional apare atunci când în aplicațiile de referință avem de a face cu date complexe;
- Limbajele de programare necesită o nouă sintaxă; mixarea; reproducerea și noile metode de acces necesită de asemenea continuarea cercetărilor; trebuie realizate caracteristici mai solide ale limbajelor de interogare; se simte nevoia continuării cercetărilor în domeniul controlului concurenței, semantica mărcilor de timp și concurenței bazate pe obiecte [2];
- Limbajul C++ nu este un limbaj prea adecvat pentru implementarea bazelor de date, întrucât prezintă probleme legate de mecanica definirii atributelor, mecanica referirii la obiecte într-un mod sistematic. Totodată, SGBD actuale bazate pe limbajul C++ duc lipsa unor importante funcții ale bazei de date și, pentru a compensa aceasta s-a recurs la implementări simple ale funcțiilor standard ale sistemelor de gestiune, cum ar fi: înregistrarea în jurul a tranzacțiilor pentru refacerea prin derulare înainte; un monitor multifir al tranzacțiilor, un limbaj și un procesor de in-

tegrări [5,6];

- Piața bazelor de date orientate obiect va continua să crească, dar va rămâne încă doar o fracțiune din piața tradițională a bazelor de date [25.33];

- Se apreciază că SGBDR dețin ponderea cea mai mare pe piața bazelor de date. Însă ca perspectivă ele vor coexista încă mult timp împreună cu SGBDOO.

Bibliografie

1. Atkinson M., Bancilhou F. Ș.a. – Object Oriented Database System Manifest. In Proc. 1 st Int. Conf. Deductive and Object Oriented Database, Kyoto, Japan, 1989.
2. R.G.G. Cattell – Object Data Management. Reading Mss., Ed. Addison Wesley, 1994.
3. Codd E.F. – Is you DBMS really relational? Computerworld, 14 oct. 1985.
4. Codd E.F. – Does your DBMS run by teh rules? Computerworld, 21 oct. 1985.
5. C. J. Date – An Introduction to Database Systems. Eighth Edition, Pearson – Addison Wesley, 2004.
6. Thomas Connolly, Carolyn Begg – Database to Design, Implementation and Management. Fourth Edition, Ed. Addison Wasley, 2005.