RDBMS vs. OODBMS

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

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 şi deosibirilor dintre modelarea relaţională şi modelarea orientată obiect a datelor este semnificativă şi de mare importanţă atât pentru proiectanţi de baze de date cât şi pentru utilizatori.

Proiectanţii, cunoscând foarte bine modelul relaţional şi apoi sesizând asemănările şi deosebirile dintre cele două moduri de abordare a modelării datelor, vor putea valorifica şi folosi experienţa dobândită anterior ca bază substanţială pentru înţelegerea şi însuşirea metodologiei de proiectare a bazelor de date orientate obiect. Totodată, prin cunoaşterea asemănărilor şi deosebirilor dintre cele două moduri de abordare apare posibilitatea conversiei unui model relaţional în obiectual şi invers. De altfel, o astfel de practică a regăsirii în mod frecvent.

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

A. Regulile de evaluare a unui SGBDR

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

E.F. Codd, din dorinţa de a menţine nealterat modelul relaţional, a elaborat 13 reguli după care ar putea fi apreciat orice SGBD dacă este cu adevărat sau nu relaţional [3, 4]. În cele ce urmează, din considerente de lipsă de spaţiu, vom recurge mai mult la o simplă enumerare a acestora.

Regula O – Regula fundamentală

Un SGBD pentru a fi considerat relaţional el trebuie să fie capabil să gestioneze în întregime bazele de date prin capacităţile sale relaţionale.

Regula amintită accentează un aspect esenţial şi anume că un SGBD nu trebuie să recurgă la nici un fel de operaţii neraleţionale pentru a realiza sarcinile ce îi revin cu privire la definirea, utilizarea şi manipularea datelor.

Regula 1 – Regula de nonsubversiune

Dacă un sistem relaţional are un limbaj de nivel scăzut, acel nivel scăzut nu poate fi utilizat pentru a submina sau ocoli regulile de integritate şi constrângerile exprimate în limbajul relaţional 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 cunoştinţa utilizatorilor sau administratorului bazei de date.

Regula 2 – Reprezentarea informaţiilor

La nivel logic, toate informaţiile dintr-o bază de date relaţională sunt reprezentate într-un singur mod şi anume prin valorile din tabele.

O astfel de regulă impune ca toate informaţiile, chiar şi meta-datele, conţinute în dicţionarul de sistem să fie stocate ca relaţii şi gestionate de către aceleaşi funcţii operaţionale care ar fi utilizate pentru întreţinerea datelor.

Regula 3 – Reactualizarea vederilor

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

Prin această regulă se stabileşte 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 deşi Codd precizează o astfel de regulă, practic nici un sistem nu acceptă cu adevărat această caracteristică, deoarece nu au fost încă descoperite condiţiile pentru identificarea tuturor vederilor care pot fi teoretic reactualizate.

Regula 4 – Tratarea sistematică a valorilor null

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

Regula 5 – Independenţa de integritate

Constrângerile de integritate specifice unei anumite baze de date relaţionale trebuie să poată fi definite în sublimbajul relaţional de date şi stocate în catalog, sau în programele de aplicaţie.

Regula 6 – Accesul garantat

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

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

Descrierea bazei de date este reprezentată la nivel logic în acelaşi mod ca şi datele obişnuite, astfel încât utilizatorii autorizaţi pot aplica la interogarea acesteia acelaşi limbaj relaţional ca cel aplicat datelor curente.

Regula 8 – Sublimbaje de date cuprinzătoare

Un sistem relaţional poate accepta mai multe limbaje şi diverse moduri de utilizare a terminalelor. Totuşi, trebuie să existe cel puţin un limbaj ale cărui instrucţiuni să poată exprima următoarele chestiuni:
(1) definirea datelor, (2) definirea vederilor, (3) manipularea datelor, (4) constrângerile de integritate, (5) autorizarea, (6) limitele tranzacţiilor (begin, commit şi rollback).

Regula 9 – Operaţii de inserare, reactualizare şi ştergere de nivel înalt

Capacitatea de tratare a unei relaţii de bază sau a unei relaţii derivate (adică o vedere) ca pe un singur operand se aplică nu numai regăsirii de date, ci şi inserării, reactualizării şi ştergerii acestora.

Regula 10 – Independenţa fizică de date

Programele de aplicaţie şi activităţile 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 – Independenţa logică de date

Programele de aplicaţie şi activităţile de la terminal rămân logic intacte, ori de câte ori sunt efectuate în tabelele de bază orice fel de modificări privind păstrarea informaţiilor, care, teoretic, permit deteriorarea acestora.

Regula 12 – Independenţa de distribuţie

Sublimbajul de manipulare a datelor dintr-un SGBD relaţional trebuie să permită programelor de aplicaţie şi înregistrărilor să rămână aceleaşi din punct de vedere logic, dacă şi ori de câte ori datele sunt centralizate sau distribuite fizic.

B. Reguli de evaluare a unui SGBDOO

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

1. Posibilitatea definirii structurilor complexe.

O astfel de cerinţă presupune capacitatea de a defini structuri complexe plecând de la tipuri de date atomice folosind o serie de tipuri de constructori, iar aceştia 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 şi formele TUPLE (SET ( )) şi TUPLE (LIST ( )).
Dintre cei mai folosiţi constructori amintim:

- ROW (n_1t_1 , . . ., n_nt_n), unde un tip reprezintă un rând sau tupleu de *n* câmpuri au câmpurile n_1, . . ., n_n de tipurile t_1 , . . ., t_n;
- ARRAY: Tipurile array suportă o metodă „array index" pentru a permite utilizatorilor să acceseze articolele array (colecției);
- seturi şi multiseturi (Sts and multisets).

Obiectele set pot fi comparate utilizând setul de metode tradițional <, ≤, =, ≥, >.

Totodată, două obiecte set (având elementele de acelaşi tip) pot fi combinate pentru a forma un nou obiect folosind operatorii $\cup, \cap$ şi – (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 şi respectiv excluderea unui element din listă. Mai există o serie de operatori, cum ar fi operatorii agregați (COUNT, SUM, AVG, MAX, MIN) şi 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 şi 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 şi datele pot fi definite publice (+), protejate (≠) sau private (-).

4. Tipuri şi/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 şi le defineşte implementarea lor. Invers, nu e necesar să fie două forme diferite de a exprima tipuri şi clase şi astfel e posibil să exprimăm un concept în contextul celuilalt.

5. Clase şi/sau tipuri de ierarhii, în contextul moştenirii, evidențiază faptul că un SGBD trebuie să asigure ca un subtip sau subclasă să poată moşteni atributele şi 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 şi să ofere facilități în ceea ce priveşte operarea şi gestionarea unor volume mari de date (baze de date de mari dimensiuni).

11. Asigurarea accesului concurent la aceleaşi 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 şi folositoare dar nu esențiale pentru un SGBDOO, dintre acestea enumerăm: moştenirea multiplă, distribuția datelor într-o rețea, posibilitatea verificării unui program în timpul compilării, managementul tranzacțiilor şi mecanisme pentru ma-

nagementul versiunilor.

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

Deosebirea esențială între cele două tipuri de modelare a datelor o reprezintă încapsularea în obiect atât a stării cât şi a comportamentului acestuia, în vreme ce modelul Entitate – Relație evidențiază numai starea şi nimic despre comportament.

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

Tabelul 1. Compararea BDOO şi BDR sub aspectul conceptelor folosite

Modelul orientat pe obiecte	Modelul relațional	Diferențe
Obiect	Entitate	Obiectul precizează şi comportamentul
Clasa de obiecte	Tipuri de entități	Clasa de obiecte include şi comportamentul comun obiectelor din clasa respectivă
Ierarhia de clase	Schema bazei de date	Ierarhia de clase implică moştenirea 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 moştenirea include atât starea cât şi comportamentul
Mesaje / interfață	Nu există	
Încapsulare	Nu există	
Identificatorul de obiect (OID)	Cheie primară	În modelul relațional dacă nu se defineşte cheia primară sistemul generează în mod automat un identificator

Deosebirile dintre modelul BDOO şi modelul BDR pot fi evidențiate şi sub aspectul obiectivelor urmărite sau prin alte caracteristici, aşa cum se poate observa din tabelul 2.

Tabelul 2. Compararea BDOO şi BDR sub aspectul obiectivelor urmărite

BDOO	BDR
• Obiective principale: încapsularea şi 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 şi modificate fizic fără a afecta modul de folosire.
• BDOO stochează date şi metode.	• BDR stochează numai date.
• Încapsularea: datele pot fi folosite numai prin metodele claselor.	• Partiționarea datelor: datele pot fi partiționare în funcție de cerințele şi 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.	• Simplicitate: utilizatorii percep datele sub formă de coloane, rânduri / tupluri şi tabele.
• Date înlănțuite. Datele pot fi înlănțuite aşa încât metodele claselor oferă performanțe sporite. Datele structurate precum şi 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 şi metodelor sunt realizate prin încapsulare şi moştenire. Moştenirea 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 şi nu în fază de dezvoltare a aplicațiilor.
• Optimizarea claselor: datele pentru un obiect	• Performanța BDR este legată de nivelul de complexi-

BDOO	BDR
pot fi interlegate şi stocate împreună, astfel încât ele pot fi accesate împreună de mecanismul de acces.	tate a structurii datelor.
• Modul conceptual consistent: modelele folosite pentru analiză, proiectare, programare şi accesul bazei de date sunt similare. Conceptele aplicaţiilor sunt în mod direct reprezentate prin clasele de obiecte.	• Model conceptual diferit: modelul structurii datelor şi acces reprezentat prin tabele şi JOIN-uri este diferit de cel în analiză, proiectare şi programare. Proiectul trebuie să fie convertit în tabele relaţionale şi acces conform SQL.

Concluzii

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

- Bazele de date relaţionale au ca obiectiv principal asigurarea independenţei datelor. Datele normalizate sunt separate de prelucrări iar prelucrările corespunzătoare satisfacerii cerinţelor informaţionale nu este obligatoriu să fie în totalitate predefinite, deci se acceptă şi cerinţele ad-hoc;
- Bazele de date orientate direct au ca obiectiv principal încapsularea, fiind stocate împreună datele şi metodele. Ele sunt inseparabile. Se spune că avem de a face cu independenţa claselor nu cu independenţa datelor;
- Nevoia unui SGBDOO şi nu unul relaţional apare atunci când în aplicaţiile de referinţă avem de a face cu date complexe;
- Limbajele de programare necesită o nouă sintaxă; mixarea; reproducerea şi 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 concurenţei, semantica mărcilor de timp şi concurenţei 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 funcţii ale bazei de date şi, pentru a compensa aceasta s-a recurs la implementări simple ale funcţiilor standard ale sistemelor de gestiune, cum ar fi: înregistrarea în jurul a tranzacţiilor pentru refacerea prin derulare înainte; un monitor multifir al tranzacţiilor, un limbaj şi un procesor de integrări [5,6];
- Piaţa bazelor de date orientate obiect va continua să crească, dar va rămâne încă doar o fracţiune din piaţa tradiţională a bazelor de date [25.33];
- Se apreciază că SGBDR deţin ponderea cea mai mare pe piaţa bazelor de date. Însă ca perspectivă ele vor coexista încă mult timp împreună cu SGBDOO.

Bibliografie

1. Atkinson M., Bancilhou F. Ş.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. Adisson 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.