MODELUL DE DATE OBIECT-RELAŢIONAL ÎN SQL

CEBAN CRISTIAN

Universitatea Tehnică a Moldovei

Abstract: *În articolul dat este descris modelul de date obiect-relaţional (Object-Relational Model), esenţa lui, tipurile de date specifice. Sunt descrise proprietăţile obiecte-relaţionale, au fost menţionate tipurile de date construite interne (build-in) , linie (rând) , definite de utilizator (UDT).*

Cuvinte cheie: *obiect-relaţional, SQL, build-in , rând, UDT, proprietăţi, tabele tipizate.*

1. Modelul de date obiect-relaţional

Modelul de date obiect-relaţional (Object-Relational Model) reprezintă extinderea modelului relaţional cu caracteristici ale modelului obiect, extindere necesară pentru realizarea bazelor de date care definesc şi prelucrează tipuri de date complexe. În esenţă, modelul obiect-relaţional păstrează structurarea datelor în relaţii (reprezentate ca tabele), dar adaugă posibilitatea definirii unor noi tipuri de date pentru domeniile de valori ale atributelor.

Tipurile de date definite de utilizator pot fi extinse prin mecanismul de moştenire şi pentru fiecare tip sau subtip se pot defini metode pe care le pot executa obiectele de acel tip. În general, dezvoltarea sistemelor de gestiune a bazelor de date obiect-relaţionale (SGBDOR) se realizează prin extinderea sistemelor relaţionale, de cele mai multe ori în mod gradat, adăugându-se de la o versiune la alta cât mai multe caracteristici posibile ale modelului obiect şi păstrând în continuare toate caracteristicile modelului relaţional.

Mai mulţi dintre principalii producători de sisteme de gestiune (Oracle, Informix, IBM şi Microsoft) au extins în acest mod sistemele lor relaţionale pentru a deveni sisteme obiect-relaţionale. Este o tendinţă firească, dat fiind că prin aceasta se păstrează toată experienţa şi rezultatele obţinute cu sistemele relaţionale şi se pot dezvolta şi aplicaţii complexe, obiect-relaţionale. Standardele limbajelor de programare pentru sistemele de gestiune obiect-relaţionale sunt extensii ale standardului SQL (ca de exemplu, versiunea din anul 1999, denumită SQL3). Stonebraker denumeşte sistemele de gestiune a bazelor de date obiect-relaţionale ca fiind sisteme de baze de date universale.

2. Proprietăţi obiect-relaţionale în standardul SQL

2.1. Tipuri construite interne (build-in)

Din faza iniţială a existenţei sale, SQL a permis utilizarea tipurilor atomice pentru definirea coloanelor, variabilelor şi parametrilor. Tot acolo a fost introdusă o clasă de tipuri de date, numite tipuri construite. Acestea sunt tipuri definite de utilizator (UDT), tipuri referinţă, tipuri linie (rând), precum şi tipuri colecţie. Tipurile construite pot conţine mai mult decât o valoare. Din cele patru componente ale familiei tipurilor construite, ultimele două (tipurile rând şi tipurile colecţie) sunt interne (build-in), iar primele două (UDT şi referinţă) pot fi folosite de utilizator pentru a defini, prin extindere, tipuri noi, pe lângă cele interne.

2.2. Tipuri linie (rând)

În accepţiunea obişnuită folosită în contextul modelului relaţional, o linie (rând) este o colecţie nevidă de valori, în care tipul fiecărei valori corespunde unei definiţii de coloană dintr-o tabelă. După cum ştim, o tabelă relaţională convenţională (relaţie) este formată din linii (rânduri) cu proprietatea că fiecare valoare de coloană, în fiecare linie, trebuie să fie atomică, aceasta fiind definiţia formei normale unu (FN1). Standardul SQL lărgeşte cerinţele FN1, prin introducerea tipului linie (rând), care admite ca o linie să conţină o altă linie, aceasta fiind introdusă ca o valoare de coloană. Se observă că tipul linie (rând) este asemănător cu structurile de înregistrare din limbajele de programare şi cu tipul *struct* din limbajul obiectual ODL.

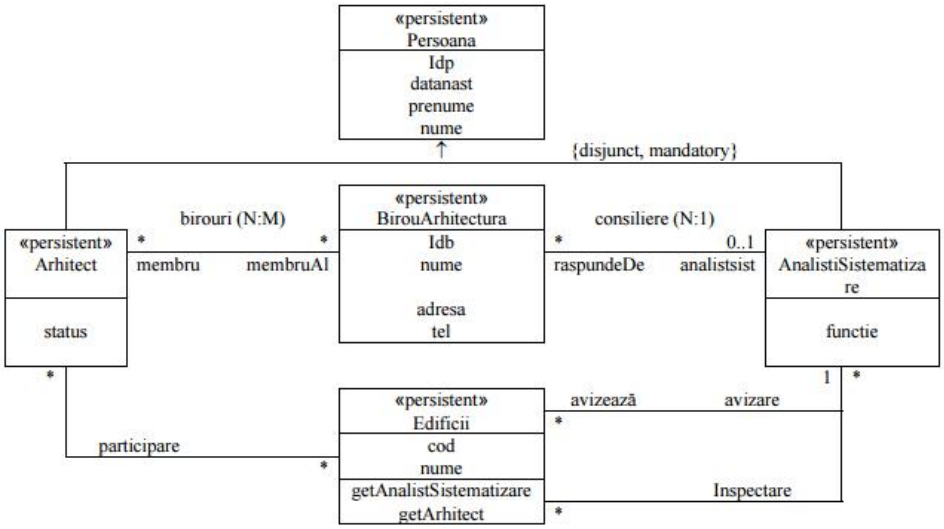

Figura 1 – Diagrama bazei de date

CREATE TABLE birouArhitectura
(
idb VARCHAR(10),
nume VARCHAR(50) NOT NULL,
adresa ROW (strada VARCHAR(30), cladirea VARCHAR(5), camera VARCHAR(5)),
AnalistSistematizare VARCHAR(11) REFERENCES arhitecti(Idp),
PRIMARY KEY (Idb)
);

Constructorul de linie (rând) se foloseşte pentru a atribui valori câmpurilor unei linii. Valorile folosite de constructorul de linie pot fi o listă de valori, sau rezultatul unei interogări. În fiecare din aceste cazuri, tipurile valorilor trebuie să corespundă tipurilor de câmp folosite în definiţia tipului linie. De exemplu:

INSERT INTO birouArhitectura
VALUES
('BA101', 'Birou Arhitect Univ', ROW('Spaiul Independentei', 'Cladire centrala', 'cam 301'), 'DA110');

În comanda de mai sus adresa biroului de arhitectură este reprezentată ca o coloană în tabelă, însă această adresă este o valoare nonatomică formată din trei valori de şiruri de caractere separate. Deasemenea, DA are sensul de Director Arhitect.

Valorile unui tip (rând) pot fi extrase folosind notaţia „cu punct" pentru a realiza accesul la câmpurile individuale, care sunt părţi ale coloanei. De exemplu:

SELECT b.adresa.strada, b.adresa.cladirea, b.adresa.camera
FROM birouArhitectura b
WHERE b.nume = 'Birou Arhitect Univ';

2.3. Tipuri definite de utilizator (UDT)

Termenul UDT este standard în SQL, având acelaşi înţeles ca termenul „tip abstract de date". În legătură cu acest ultim concept, ne reamintim că utilizatorul defineşte tipuri noi care au o anumită formă a structurii interne, fiind de asemenea definite metode care descriu comportarea acelor tipuri. Reprezentarea internă a unui tip este încapsulată de comportarea acestuia, ceea ce înseamnă că implementarea internă a tipului de date este ascunsă faţă de lumea exterioară. Această implementare se poate schimba, fără a influenţa modul în care un utilizator interacţionează cu tipul de date.

În standardul SQL, tipurile definite de utilizator sunt însă întrucâtva diferite faţă de definiţia strictă a tipurilor abstracte de date. Diferenţa provine, în special, din faptul că atât toate atributele interne ale unui UDT, cât şi metodele asociate, sunt publice. Aceste elemente nu pot fi „mascate" ca fiind protejate sau private, aşa cum se procedează în limbaje ca Java sau C++. Folosirea atributelor publice şi a metodelor dă posibilitatea ca structura internă a unei instanţe a tipului introdus de utilizator să fie integrată printr-un limbaj ca SQL. Virtutea principală a UDT constă în faptul că ele permit proiectanţilor de baze de date să definească tipuri noi de date, orientate către aplicaţie, în afară de tipurile atomice şi construite din categoria „build-in", iar apoi să folosească noile tipuri în definirea de tabele (relaţii). Desigur, cu ajutorul UDT pot fi create tabele în mediul relaţional.

2.4 Tabele tipizate

În standardul SQL, o instanţă a unui UDT este o valoare. Pentru a crea conceptul de obiect, ca în tehnologia bazelor de date orientate obiect, trebuie folosit un UDT împreună cu o tabelă tipizată. Tabela tipizată este o formă nouă de tabelă în standardul SQL, asociată întotdeauna cu un anumit tip structurat. Pentru fiecare atribut al tipului structurat pe care se sprijină, o tabelă tipizată are o coloană. În plus, va avea o coloană cu „auto-referire", care conţine un identificator unic de obiect pentru fiecare linie (rând) al tabelei. Acest identificator se numeşte referire. Când pentru definirea unei tabele tipizate se foloseşte un tip structurat, o instanţă a acestui tip este văzută ca un obiect, al cărui identificator este dat de coloana cu „auto referire". Spre deosebire de identificatoarele de obiecte folosite în bazele de date orientate obiect, un identificator de obiect este unuic numai în contextul unei tabele tipizate specifice. Deci, două tabele tipizate pot avea linii cu valori cu „auto-referire" identice.

2.5. Tablouri de colecţii

Valorile nonatomice din schemele relaţionale pot fi reprezentate şi prin tipul colecţie. În general, o colecţie poate fi o structură de date din categoria mulţimilor, listelor, multiset-urilor sau tablourilor. Standardul SQL permite, în forma sa actuală, numai tipul tablou. O coloană a unei tabele poate fi definită (specificată) ca un tablou, adăugând la tipul de date al coloanei cuvântul cheie *array*. La rândul său, acest cuvânt cheie este urmat de numărul maxim de elemente pe care le poate conţine, înscris între paranteze drepte. Accesul la prima poziţie dintr-o tabelă este realizat cu o valoare de index de unu.

3. Concluzii

Modelul obiect relaţional (Object Relational Model) este următorul mare val în dezvoltarea şi întreţinerea bazelor de date. Construcţia se poate realiza dezvoltând sistemul relaţional prin adăugarea caracteristicilor obiectuale necesare sau pornind de la un sistem orientat obiect şi adăugând caracteristicile relaţionale. Aşadar, sistemele de gestiune a bazelor de date obiect-relaţionale devin un sisteme de baze de date universale.

Bibliografie

1. Mircea Petrescu. Proprietăţi obiectual-relaţionale în standardul SQL. Universitatea Politehnica Bucureşti.
2. Modelul relațional-obiectual. [Resursa electronică]:-Regim de acces: http://www.scritub.com/stiinta/informatica/sql/Modelul-relationalobiectual54651.php
3. Baze de date. Curs pe portalul UTCB. [Resursa electronică]:-Regim de acces: http://moodleaia.utcb.ro/course/view.php?id=6
4. Tipuri de date în SQL. Wikipedia. [Resursa electronică]:-Regim de acces: https://ro.wikipedia.org/wiki/SQL#Tipuri_de_date_.C3.AEn_SQL