Capitolul 3. Standarde şi dialecte SQL. Tipuri de date. Creare tabelelor şi modificarea conținutului

De obicei, noțiunea de limbaj de programare duce cu gândul la Pascal, C, C++, Basic, Fortran, COBOL, PL/1, Algol, Java etc. Inițiații ştiu că mai există limbaje de asamblare şi limbaje-maşină. Toate aceste limbaje sunt generale, în sensul că permit realizarea de module ale oricărui strat dintr-o aplicație file-server, client/server sau web (vezi paragraful 1.2). Este foarte posibil ca meniurile unei aplicații să fie realizate în Visual Basic, o parte dintre formulare în C++, iar o altă parte în Visual Basic.

Cu bazele de date, am văzut că lucrurile au stat, încă de la începuturile acestora, un pic altfel. Niciunul dintre limbajele de uz general nu stă grozav în materie de comenzi DDL (Data Definition Language), DML (Data Manipulation Language) sau DCL (Data Control Language).

Astfel că toate SGBD-urile, servere BD sau nu, au dispus (şi dispun în continuare) de un limbaj propriu „înțesat" de comenzi DDL, DML şi DCL. Profunda incompatibilitate între limbajele numeroaselor SGBD-uri a atras preocupări majore pentru elaborarea unor standarde care să armonizeze „dialectele" acestora.

Dintre limbajele special dedicate bazelor de date, propuse din anii 60 încoace, unul a reuşit să se impună în toate serverele importante – limbajul SQL (al cărui nume este acronimul de la Structured Query Language). Succesul său fără echivalent în lumea bazelor de date l-a făcut pe Michael Stonebraker să califice SQL-ul, într-un acces de entuziam, drept limbaj intergalactic pentru date (*intergalactical dataspeak*)[1].

După Groff şi Weinberg, principalele atuuri ale SQL sunt[2]:

- Independența de producător, nefiind o tehnologie proprietară. Prin urmare, modificarea, într-o aplicație, a serverului BD, nu este atât de traumatizantă.

---

[1] [Stonebraker s.a.1998]

[2] [Groff & Weinberg 2002]

- Portabilitate între diferite sisteme de operare: astăzi SQL este implementat pe servere BD ce rulează pe PC-uri, mainframe-uri etc.
- Este standardizat, având recunoaştere planetară (sună cam pompos, dar afirmaţia este reală!).
- Are sprijinul nu numai organismelor de standardizare, dar şi al marilor firme producătoare de instrumente pentru gestionarea BD: IBM, Oracle, Microsoft.
- "Filosofia" sa se bazează pe modelul relaţional de organizare a datelor. Pentru unii acest lucru ar putea fi un dezavantaj, însă structura tabelară a relaţiilor rămâne uşor de înţeles şi deprins de către utilizatori.
- Este un limbaj de nivel înalt, cu sintaxă apropiată de limba engleză, de aici relativa facilitate în deprinderea comenzilor.
- Permite formularea de răspunsuri la numeroase întrogări simple, ad-hoc, neprevăzute iniţial. Acesta rămâne încă un atu important al BD relaţionale în competiţia cu cele pur obiectuale.
- Constituie suportul „programatic" pentru accesul la BD. Aproape orice modul al unei aplicaţii cu BD conţine comenzi SQL incluse în programe redactate în cele mai diverse limbaje, de la PHP la Java şi C++.
- Permite imagini multiple asupra datelor din bază, fără a afecta cu nimic corectitudinea şi integritarea BD.
- Este un limbaj relaţional complet. Pentru cei doi autori, noţiunea de completitudine se referă la faptul că, deşi a pornit de la interogarea BD, SQL se foloseşte pentru o largă categorie de operaţiuni: crearea de obiecte din bază, actualizarea datelor, crearea conturilor pentru utilizatori, acordarea de drepturi etc.
- Permite definirea dinamică a datelor, în sensul modificării structurii bazei chiar în timp ce o parte din utilizatori sunt conectaţi la BD.
- Constituie un excelent suport pentru implementarea arhitecturilor client-server şi web.

Nu există astăzi lucrări sau publicaţii în domeniul SGBD-urilor care să nu prezinte mecanismul de lucru al acestui limbaj sau ultimele produse/tendinţe din lumea SQL. Impactul său este profund. SQL este, pe de o parte, unul dintre "responsabilii" noii apropieri a non-informaticianului de datele sale, iar, pe de altă parte, pentru profesionişti, reprezintă nucleul dezvoltării aplicaţiilor ce utilizează bazele de date.

Deşi este referit, în primul rând, ca un limbaj de interogare, SQL este mult mai mult decât un instrument de consultare a bazelor de date, deoarece permite, în egală măsură:

- definirea datelor;
- consultarea BD;
- manipularea datelor din bază;
- controlul accesului;
- partajarea bazei între mai mulţi utilizatori ai acesteia;
- menţinerea integrităţii BD.

Setul comenzilor de bază care îndeplinesc aceste funcţionalităţi este cuprins în tabelul 3.1.

Tabel 3.1 Cele mai importante comenzi SQL

Comandă	**Scop**
Pentru manipularea datelor	
SELECT	Extragerea datelor din bază
INSERT	Adăugarea de noi linii într-o tabelă
DELETE	Ştegerea de linii dintr-o tabelă
UPDATE	Modificarea valorilor unor atribute
Pentru definirea bazei de date	
CREATE TABLE	Adăugarea unei noi tabele în baza de date
DROP TABLE	Ştergerea unei tabele din bază
ALTER TABLE	Modificarea structurii unei tabele
CREATE VIEW	Crearea unei tabele virtuale
DROP VIEW	Ştergerea unei tabele virtuale
Pentru controlul accesului la BD	
CREATE USER	Crearea contului unui utilizator
ALTER USER	Modificarea proprietăţilor contului unui utilizator
DROP USER	Ştergerea contului unui utilizator
GRANT	Acordarea unor drepturi la obiecte din bază pentru un cont de utilizator sau grupuri (roluri) de utilizatori
REVOKE	Revocarea unor drepturi
Pentru controlul tranzacţiilor	
COMMIT	Marchează sfârşitul „corect” al unei tranzacţii
ROLLBACK	Abandonează tranzacţia în curs

3.1. Standardizarea SQL

În privinţa standardizării SQL, lucrurile sunt destul de încâlcite. Putem vorbi de standarde SQL, adică SQL-89, SQL-92, SQL:1999, SQL:2003, dar, la fel de bine, dacă ţinem cont că numărul alocat de ISO este acelaşi – 9075 -, putem vorbi, cel puţin la fel de îndreptăţiţi, de versiunile SQL-89, ... ale standardului SQL. Pentru a menţine deruta, noi vom folosi ambele exprimări.

O altă sursă de zăpăceală este că există în istoria SQL versiuni intermediare sau revizuiri minore ale standardului, aşa încât la un inventar ceva mai atent am putea completa pomelnicul: SQL-86, SQL-89, SQL-92, SQL:1999, SQL:2003, SQL:2006, SQL:2008. Mutiplicarea (sub)versiunilor standardului are şi un efect benefic: nu ne mai simţim obligaţi să le ţinem minte pe toate. La drept vorbind, senzaţia de

degringoladă vis-a-vis de greu numărabilele versiuni se datorează faptului că standardul SQL nu mai este, începând cu varianta 1999, monolitic, ci compus dintr-o serie de părți pe care le vom evoca peste câteva pagini. O variantă a standardului poate să însemne, de fapt, apariția sau modificarea substanțială a unei părți-două, restul rămânând aproape intact.

3.1.1. Apariția SQL-ului

La începutul anilor '70 piața era dominată de SGBD-urile care funcționau după principiile modelelor ierarhice şi rețea. Liderul autoritar (concurenții erau nesemnificativi în acea perioadă) al bazelor de date era firma IBM, care domina copios atât piața de hardware (celebra serie System/360 de mainframe-uri), cât şi pe cea de software.

În cele două articole ale inventatorului modelului relațional [Codd 1969] [Codd 1970], acesta sugera "adoptarea unui model relațional pentru organizarea datelor [...] care să permită punerea la punct a unui sub-limbaj universal pentru gestiunea acestora, sub-limbaj care să fie, în fapt, o formă aplicată de calcul asupra predicatelor". După cum vom vedea în capitolul următor, există două categorii majore de limbaje de interogare, una bazată pe algebra relațională şi alta bazată pe calcul relațional. Codd a mizat pe a doua categorie (la urma urmei, modelul relațional este bazat pe predicate), însă câştig de cauză pe piață au avut limbajele algebrice (SQL).

Sunt poate două dintre motivele care explică ciudățenia ca IBM-ul să nu includă în echipa de dezvoltare a SQL-ului tocmai pe părintele modelului relațional. În plus, poate că şi neincluderea lui Codd în proiectul SQL a accentuat înverşunarea lui Codd şi relaționiştilor la adresa limbajului.

Un prim moment important în naşterea SQL îl constituie lansarea proiectului System/R de către firma IBM, eveniment ce a avut loc în 1974[3]. Tot în 1974 Chamberlin şi Boyce au publicat un articol[4] în care este prezentat un limbaj structurat de interogare, denumit SEQUEL (Structured English as QUery Language). SEQUEL a fost gândit de fapt ca o interfață (API – Application Program Interface) a aplicațiilor cu SGBD-ul System R[5].

---

[3] Vezi [Astrahan s.a. 1976]

[4] [Chamberlin&Boyce 1974]

[5] [Melton & Simon 2002], p.24

În 1975 Chamberlin, Boyce, King şi Hammer redactează o lucrare dedicată sub-limbajului SQUARE[6], asemănător SEQUEL-ului, dar care utilizează expresii matematice şi nu cuvinte din limba engleză. Autorii celor două studii au demonstrat că limbajele SEQUEL şi SQUARE sunt complete din punct de vedere relaţional. În 1976 o echipă de autori condusă de Chamberlin elaborează o nouă lucrare[7] în care se face referire la SEQUEL 2, acesta fiind declarat limbaj de interogare al SGBD-ului System /R al firmei IBM. Chamberlin schimbă denumirea SEQUEL în SQL - Structured Query Language (Limbaj Structurat de Interogare)[8] în 1980, dar şi astăzi mulţi specialişti pronunţă SQL ca pe predecesorul său. Don Chamberlin a lucrat la IBM până în 2003 (din 2003 este IBM Fellow), fiind implicat în dezvoltarea mai multor versiuni ale SGBD-ului DB2, iar în ultimii ani este co-autor la documente şi standarde XML (XQUERY).

3.1.2. SQL-86 şi SQL-89, adică SQL1 şi SQL1.1

Stardardele internaţionale se redactează sub egida ISO (International Organization for Standardization). În materie de SQL motorul standardizării a fost şi este ANSI (American National Standards Institute).

Încercând să răspundă solicitărilor privind standardizarea unui limbaj de lucru cu bazele de date, ANSI a încredinţat această sarcină comitetului X3H2 în anul 1982. Comitetul pentru bază de date X3H2 (unul din numeroasele consilii tehnice ale X3) a fost (şi este, dar sub un alt nume, după cum o să vedem ceva mai jos) format din experţi independenţi şi din reprezentanţi ai firmelelor importante din domeniul bazelor de date. Figura 3.1 ilustrează poziţia X3H2 în organismele de standardizare în anii 80 şi 90[9].

În această schemă, un standard nou începe cu o schiţă, *proiect de lucru* (working draft) întocmit de comitet (X3H2), proiect ameliorat până când membrii convin că este aproape de finalizare. Proiectul de lucru devine *proiect al comitetului* (committee draft) care este transmis comunităţii profesionale şi firmelor pentru comentarii şi propuneri. După preluarea observaţiilor şi propunerilor, proiectul comitetului este publicat sub titulatura *proiect internaţional de standard* şi, în final, *standard internaţional*.

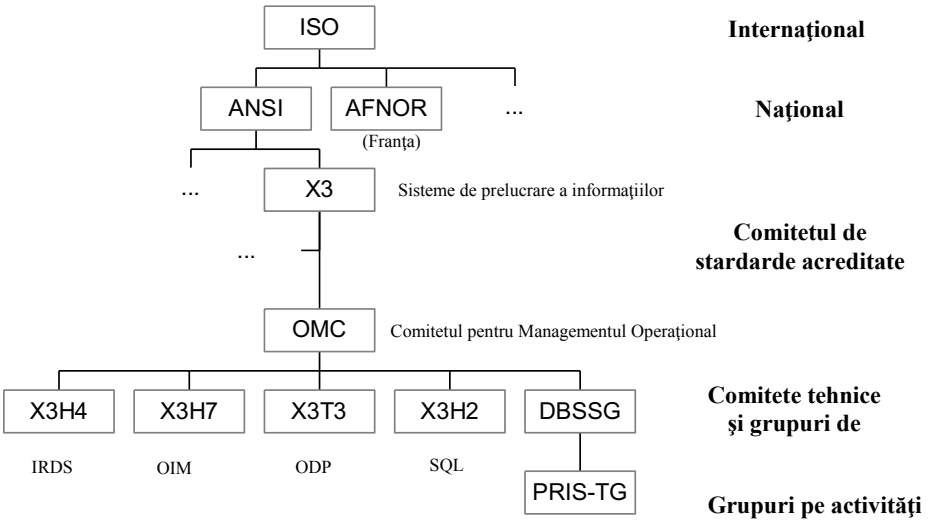
---

6 [Boyce s.a. 1975]

7 [Chamberlin s.a. 1976]

8 Se pare că a fost şi o problemă de copyright asupra sintagmei SEQUEL – vezi [Hernandez&Viesca 2000], p.52

9 Preluare din [Fortier 1999], p.4



Iată semnificaţia câtorva dintre abrevieri:
ISO – International Organisation for Standardization
ANSI – American National Standard Institute
AFNOR – Association Francaise de Normalisation
OMC – Operational Management Committee
DBSSG – Database Systems Study Group
PRIS-TG – Predictable Real-time Information Management Task Group

Figura 3.1. Structura organismelor de standardizare în domeniul bazelor de date în anii `80 şi începutul anilor `90

Comitetul X3H2 a primit responsabilitatea elaborării unui standard pentru un limbaj dedicat bazelor de date în anul 1982. Rapid (în circa doi ani), au fost elaborate câteva versiuni ale standardului, bazate copios pe specificaţiile IBM SQL/DS şi DB2. Fiind considerat a fi mult prea impregnat de IBM, în 1984 standardul SQL a fost reproiectat pentru a-i asigura un minim de generalitate.

ANSI publică în 1986 standardul *SQL ANSI X3.135-1986*, standard ce utilizează, pentru reprezentarea sintaxei, forma Backus-Naur. Este un standard care continuă să se bazeze, într-o mare măsură, pe "dialectul" SQL al IBM, şi se întinde pe aproximativ 100 de pagini. Organizaţia Internaţională pentru Standarde

(ISO) a adoptat propriul document, aproape identic cu ANSI SQL-86, pe care l-a publicat în 1987 ca *ISO 9075-1987 Database Language SQL*.

SQL-86 defineşte comenzile de bază ale SQL, inclusiv pentru crearea de tabele şi tabele virtuale (CREATE TABLE, CREATE VIEW), însă nu conţine opţiuni de modificarea a structurii sau ştergere (ALTER …/ DROP…) şi nici comenzi pentru acordare şi revocare de drepturi utilizatorilor (GRANT/REVOKE)[10]. Lipsa facilităţilor privind definirea şi asigurarea restricţiilor referenţiale a generat discuţii aprinse şi critici vehemente, astfel încât, rapid, a fost publicat un set de specificaţii numit *Integrity Enhancement Feature* (elemente de ameliorare a integrităţii) prin care se pot defini chei primare şi chei străine ca elemente componente ale schemei bazei de date.

La trei ani de la publicarea SQL-86, prin revizuirea şi extinderea sa, se "naşte" SQL-89, care mai este denumit, în majoritatea lucrărilor de profil, SQL-1[11] sau SQL-89 – ANSI X3.135-1989, respectiv ISO 9075:1989. Deşi recunoscut ca fundament al multor SGBDR-uri comerciale, şi SQL1.1 şi-a atras numeroase critici. În plus, variantele comercializate de diferiţii producători, deşi esenţialmente asemănătoare, erau (şi sunt) incompatibile la nivel de detaliu. În afară de setul de specificaţii mai sus amintit, SQL-89 a inclus şi specificaţiile pentru apelul comenzilor şi funcţiilor SQL din limbaje-gazdă, precum COBOL, Fortran şi C.

Pe lângă ANSI, ale cărui standarde au cea mai largă audienţă, au fost înfiinţate şi alte organisme de standardizare SQL. X/Open este un grup de firme europene care a adoptat SQL ca nucleu al unei întregi serii de standarde menite să asigure realizarea unui mediu general pentru aplicaţii portabile, grefat pe sistemul de operare UNIX. IBM a avut un aport incontestabil la apariţia şi maturizarea SQL, fiind un producător cu mare influenţă în lumea SGBD-urilor, iar produsul său, DB2, a fost şi este un element de referinţă al SQL.

În 1989 un grup de producători de instrumente dedicate bazelor de date au format *SQL Access Group*, în vederea realizării conexiunilor dintre SGBDR-urile fiecăruia, pe baza unor specificaţii comune, din care un prim set a fost publicat în 1991 sub titulatura RDA (Remote Database Access). Specificaţiile RDA n-au reuşit să se impună pe piaţa SGBD-urilor. La insistenţele firmei Microsoft, SQL Access Group şi-a concentrat eforturile pentru elaborarea unei interfeţe-standard pentru SQL. Pe baza unui set de propuneri înaintat de companie, în 1992 au rezultat

---

10 Pentru detalii privind istoria SQL, vezi şi cuvântul înainte al lui Ken Jacobs la lucrarea [Kreines 2000] sau [Hernandez & Viesca 2000], pp.52-63

11 Eu îmi asum "notaţia" din [Kriegel & Truknov 2008], oarecum "minoritară", prin care SQL-86 este referit ca SQL1, iar SQL-89 drept SQL1.1.

specificațiile CLI (Call Level Interface). Având drept reper CLI, Microsoft a elaborat şi implementat în acelaşi an un set propriu, ODBC (Open DataBase Conectivity), care a devenit standard în materie de interfață SQL pentru accesarea diferitelor baze de date.

3.1.3. SQL-92

Pentru a umple golurile SQL1.1, ANSI şi ISO au elaborat în 1992 versiunea SQL-2, ANSI X3.135-1992 (*Database Language SQL*), respectiv ISO/IEC 9075:1992, specificațiile fiind prezentate la un nivel mult mai detaliat (dacă SQL1 se întindea pe numai 100 de pagini, SQL-92 a fost publicat în aproape 600). Dintre numeroasele facilități aduse de SQL-92, merită amintite cu deosebire: joncțiunea externă (OUTER JOIN), atribute de tip dată-oră şi de alte tipuri, raportare standardizată a erorilor, un set standard de tabele din catalog (dicționarul de date), modificarea schemei bazei (DROP, ALTER, GRANT, REVOKE), SQL dinamic, modificări şi ştergeri referențiale în cascadă, amânarea verificării restricțiilor, nivele de consistență a tranzacțiilor etc.

Standardul SQL-92 defineşte trei nivele de conformitate:

- *Entry* – intrare, de bază (opțiunile din SQL-89 corectate, plus câteva mici adăugiri);
- *Intermediate* – intermediar, ce include aproximativ jumătate dintre facilități, cum ar fi: denumirea restricțiilor, suport pentru şiruri de caractere de lungime variabile şi pentru seturi de caractere „naționale", structuri de control de tip CASE, funcții de conversie (CAST), operatorii INNER şi OUTER JOIN, SQL dinamic, operatorii ansamblişti (UNION, INTERSECT, EXCEPT)[12];
- *Full* – deplin, dedicate unor opțiuni avansate: restricții amânabile, aserțiuni, tabele temporare locale, domenii etc[13].

Fiecare producător îşi declara nivelul de conformitate al SGBD-ului în raport cu SQL-92. Spre exemplu, nucleul SQL din Oracle 8 era conform cu nivelul de bază (*entry*), dar, după declarațiile producătorului, prezenta şi multe elemente suplimentare specifice celorlalte două niveluri superioare. Certificarea nivelului de conformitate cădea în sarcina unui organism independent, *National Institute for Standards and Technology* (NIST) care utiliza *Federal Information Processing Standards*

[12] [Kriegel & Truknov 2008], p.24

[13] [Kriegel & Truknov 2008], p.24

(FIPS), FIPS PUB 127-2. Din 1997, însă, FIPS şi-a declinat implicarea în activitatea de certificare[14].

Despre SQL-92, referit uneori şi ca SQL2, se spune că este cel mai important dintre standarele SQL, ba chiar cel mai important dintre standardele dedicate bazelor de date.

3.1.4. SQL:1999

În 1997, grupul X3 al ANSI a fost redenumit în NCITS (National Committee for Information Technology Standards), iar comitetul care se ocupă de standardizarea SQL se numeşte acum *ANSI NCITS-H2 (Database)*.

Următorul standard a fost numit inițial SQL3. Cum cea mai mare parte a sa a fost publicată în iulie 1999, denumirea propusă de ANSI şi ISO este SQL:1999. Complexitatea superioară față de predecesori este sugerată şi de numărul de pagini, aproape 2000 (față de 600 ale SQL-92). Scadența finalizării sale a fost repetat amânată. Principalele orientări ale SQL:1999 au vizat transformarea acestuia într-un limbaj complet, în vederea definirii şi gestionării obiectelor complexe şi persistente. Aceasta include:

- generalizare şi specializare,
- moşteniri multiple,
- polimorfism,
- încapsulare,
- tipuri de date definite de utilizator,
- expresii privind interogări recursive şi instrumente adecvate de administrare a datelor.

Defalcarea inițială (1993) a standardului SQL operată de comitetele ANSI şi ISO a avut în vedere şapte componente[15]:

- Partea 1. *Cadru general*. Descrie fiecare parte a standardului şi conține informații comune tuturor părților.
- Partea 2. *Fundament*. Defineşte sintaxa şi semantica SQL în ce priveşte definirea şi manipularea bazei de date, inclusiv opțiuni privind gestiunea tipurilor abstracte de date.
- Partea 3. *SQL/CLI* (*Call Level Interface*): un ansamblu de funcții şi proceduri pentru conectarea bazelor de date prin SQL în medii multi-utilizator şi multi-platformă, ansamblu dezvoltat de SQL Access Group. Cea mai

[14] Vezi şi [Gorman 1997]

[15] http://www.jcc.com/SQLPages/jccs_sql.htm

faimoasă implementare a SQL/CLI este ODBC (Object DataBase Connectivity), la care firma Microsoft a avut o contribuţie decisivă.
- Partea 4. *SQL/PSM* (*Persistent Stored Modules*): specificaţii procedurale necesare în funcţii şi proceduri utilizator.
- Partea 5. *SQL/Bindings*: include *Dynamic SQL* şi *Embedded SQL* din standardul SQL92 (SQL2) care se referă la modul în care SQL este inclus în limbajele de programare ne-obiectuale. SQL/OLB (Object Language Bindings) furnizează posibilitatea de a include comenzi SQL în programe Java, pe baza JDBC (Java DataBase Connectivity).
- Partea 6. *SQL/XA*: specificaţii elaborate de X/Open şi dedicate platformei X Windows. Această parte a fost abandonată.
- Partea 7. *SQL/Temporal*: adaugă noi facilităţi privind gestiunea timpului şi datei calendaristice în SQL.

La această structurare iniţială, între timp au mai fost adăugate şi alte părţi precum[16]:
- *SQL/OLAP*. Sunt descrise funcţiile şi operaţiunile utilizate pentru prelucrări analitice, fiind publicate ca amendament la standardul SQL:1999[17].
- Partea 8. *SQL/Objects – Extended Objects*. Vizează modul în care SGBD-urile relaţionale gestionează tipurile abstracte de date în aplicaţii. Nici această componentă nu mai există astăzi, fiind transferată în întregime în Fundament.
- Partea 9. *SQL/MED (Management of External Data)*. Defineşte câteva elemente adiţionale Fundamentului pentru accesarea unor surse de date de tipul fişierelor şi bazelor de date.
- Partea 10: *SQL/OLB (Object Language Bindings)*. Este inclusă numai în standardul ANSI şi definitivată din 1998; cuprinde specificaţii privitoare la includerea frazelor SELECT în limbajul Java, fiind corespondentă unui alt standard ANSI, SQLJ (Partea 0).
- Partea 11: *SQL/Schemata*. Se referă la definirea şi extragerea informaţiilor privind schema bazei de date; este parte din Fundament, dar, în viitor, această parte va fi de-sine-stătătoare.

---

[16] Preluare din [Hernandez&Viesca 2000], pp.62-63

[17] Vezi şi [Fotache2000 – 4] şi [Fotache2000 - 5]

- *SQL Routines using the Java Programming Language*. Definitivată în 1999 şi inclusă numai în standardul ANSI, această parte se referă la modul în care secvenţe de cod Java pot fi incluse în bazele de date SQL.
- Partea 12: *SQL/Replication*. Lucrările au demarat în 2000, fiind vizată standardizarea comenzilor pentru replicarea bazelor de date.

Dintre ameliorările nelegate strict de lucrul cu obiecte, merită amintite cele referitoare la: declanşatoare (triggere) şi alte tipuri de proceduri stocate, suport pentru seturile de caractere naţionale, noi predicate în clauza WHERE (FOR ALL, FOR SOME, SIMILAR TO), tabele virtuale actualizabile, roluri pentru definirea profilelor de securitate etc. Se poate spune că, odată cu standardul publicat în 1999, SQL iese din sfera relaţionalului.

3.1.5. SQL:2003

Dominanta standardului SQL:2003 este XML. În afară de corecţia câtorva inconsecvenţe ale SQL:1999, au mai fost:

- eliminate tipurile de date BIT şi BIT VARYING;
- introduse tipurile de date BIGINT, MULTISET, XML;
- modificate câteva opţiuni DDL (CREATE TABLE LIKE şi CREATE TABLE AS) şi DML (MERGE);
- introduse câteva noi opţiuni, cum ar fi funcţiile-tabelă, secvenţe, coloane-identificator[18].

SQL:2003 cuprinde nouă părţi, opt preluate din SQL:1999, plus una dedicată XML-ului[19]:

- Cadru general – *SQL Framework* (partea 1);
- Fundament – *SQL Foundation* (partea 2);
- *SQL/CLI* (Call Level Interface) (partea 3);
- *SQL/PSM* (Persistent Stored Modules) (partea 4);
- *SQL/MED* (Management of External Data) (partea 9);
- *SQL/OLB* (Object Language Bindings) (partea 10);
- *SQL/Schemata* (partea 11);
- *SQL/JRT* (Java Routines and Types) (partea 13);
- *SQL/XML* (partea 14).

---

[18] Vezi şi [Eisenberg s.a.2004]

[19] [Kriegel & Truknov 2008], p.25, [Kulkarni 2003]

3.1.6. SQL:2008

Pe site-ul ISO a fost publicat (şi poate fi cumpărat) SQL:2008[20], elaborat, în colaborare, de ISO şi IEC (International Electrotechnical Commission). Cele două organisme internaţionale au un comisie tehnică comună dedicată exclusiv standardelor ce privesc tehnilogiile informaţionale – *JTC 1* (Joint Technical Committee 1) şi, în cadrul acesteia, subcomisia SC32 denumită *Data management and interchange*. Stardardele se aprobă dacă există o rată de acceptare de cel puţin 75% dintre organismele naţionale de standardizare.

Ca şi precedentul standard (ISO/IEC 9075:2003), versiunea ISO/IEC 9075:2008 prezintă nouă părţi, cu titulaturi identice. Noutăţile vizează partea dedicată XML, adăugarea unui nou tip BINARY, suport pentru căutarea pe bază de şabloane (*regular expressions*[21]), clasamente (FIRST n / TOP n) etc[22].

Nu aş vrea să credeţi că standardizarea SQL este un marş triumfal. Lumea teoreticienilor şi profesioniştilor în ale bazelor de date este destul de fragmentată în aprecierea virtuţilor şi slăbiciunilor stardardelor SQL. Obiecţiile sunt atât de natură conceptuală, venind dinspre tabăra teoreticienilor relaţionali (E.F. Codd, C.J. Date, H. Darwen, F.Pascal), dar şi dinspre practicieni care deplâng lipsa de conformitate SQL a serverelor BD actuale faţă de ultimele două-trei versiuni ale standardului[23].

3.2. Tipuri primitive de date SQL

La crearea „efectivă" a oricărei tabele dintr-o bază gestionată de un server BD, pentru fiecare atribut, pe lângă nume şi eventuale restricţii trebuie declarat tipul acestuia – dacă valorile sale vor fi numere, şiruri de caractere etc. Mult timp modelului relaţional i s-a imputat că poate gestiona numai date simple, adică numere, şiruri de caractere şi date calendaristice. Primele două standarde SQL erau "puse la index" pe acelaşi motiv. Începând cu SQL:1999 tipologia datelor este cu mult mai generoasă. În afară de tipurile pre-definite noi, cum ar fi tipurile INTERVAL, ARRAY, au fost introduse opţiuni care să permită utilizatorilor să-şi definească şi implementeze propriile tipuri, în funcţie de cerinţele aplicaţiilor.

---

[20] www.ansi.org

[21] Vezi capitolul 5.

[22] [Kriegel & Truknov 2008], p.25

[23] [Gulutzan 2005], [Gorman 2001]

SQL:2008-1 defineşte trei tipuri de date: tipuri *predefinite*, tipuri *construite* şi tipuri *definite de utilizator*. În următoarele capitole ne interează cu precădere tipurile din prima categorie, pe care le putem denumi *primitive* sau *clasice*, deşi în discutarea vor fi câteva trimiteri şi la tipuri din celelalte două categorii. Vom trece sub tăcere datele de tip XML, deşi sunt la modă.

3.2.1. Categoriile "clasice" de tipuri de date

Deşi există unele diferențe între dialectele SQL-urile, următoarele categorii de tipuri de date sunt cvasi-prezente încă din primele două standarde:

- Pentru numere:
  - SMALLINT: întregi - scurte;
  - INTEGER sau INT: întregi;
  - NUMERIC(p,s) sau DECIMAL(p,s) sau DEC(p,s) - reale, cu un total de p poziții, din care s la partea fracționară ,
  - FLOAT: reale, virgulă mobilă;
  - REAL: real, virgulă mobilă (cu precizie mai mică decât FLOAT, dar la nivelul de intrare este identic);
  - DOUBLE PRECISION: reale, virgulă mobilă, dublă precizie.
- Pentru şiruri de caractere:
  - CHAR(n) sau CHARACTER(n): şir de caractere de lungime (fixă);
  - VARCHAR(n) sau CHAR VARYING(n) sau CHARACTER VARYING(n): şir de caractere de lungime variabilă.
- Pentru date şi ore calendaristice:
  - DATE: dată calendaristică;
  - TIME: ora, minut, secundă, dupa caz şi fracțiuni de secundă;
  - TIMESTAMP: an, lună, zi, ora, minut, secundă.

3.2.2. Categorii recente de tipuri de date

Din SQL:1999 lucrurile s-au schimbat mult (în bine) în materie de tipuri de date. O serie de tipuri clasice au fost completate cu subtipuri, însă cele mai interesante sunt câteva tipuri noi-noute.

Numere

SQL:2003 introduce, printre altele, tipul BIGINT care, după cum îi spune şi numele, permite stocarea şi manipularea unor numere întregi de dimensiuni mai mari decât INTEGER.

Şiruri de caractere

Alături de „clasicele" CHARACTER (CHAR) şi CHARACTER VARYING (VARCHAR), din SQL:1999 apare tipul CHARACTER LARGE OBJECT (CLOB) util pentru atribute ale căror valori sunt şiruri de caractere de lungime variabilă şi suficient de mare pentru a fi exprimată în kilo (1024, mega (1048576) şi giga (1073741824) caractere.

Mai merită de amintit că toate tipurile pentru şiruri de caractere pot fi gestionate nu numai în engleză, ci şi în alte limbi, unele exotice, din punctul de vedere al caracterelor folosite şi principiilor fonetice (Hebrew, Hindu, Japanese, Chinese, Korean etc.)[24]. Pentru aceasta standardul SQL:1999 a prevăzut folosirea clauzei NATIONAL, fiind, astfel, disponibile trei (relativ) noi tipuri: NATIONAL CHARACTER (NCHAR), NATIONAL CHARACTER VARYING (NCHAR VARYING) şi NATIONAL CHARACTER LARGE OBJECT (NCLOB). Implicaţiile sunt importante atât pentru stocare, cât şi prelucrare/ordonări.

BLOB-uri

Tipul de date BINARY LARGE OBJECT a fost gândit pentru stocarea în baza de date a imaginilor, sunetelor şi secvenţelor video. Ca şi în cazul CLOB-urilor, lungimea unui atribut de tip BLOB se poate exprima în multipli de octeţi: K, M şi G. Limitele de mărime ale BLOB-urilor difera de la SGBD la SGBD. Principalul dezavantaj al acestui tip este opacitatea sa, SGBD-urile neavând implementate funcţii care să permită extragerea valorilor după anumite criterii. De exemplu, am fi interesaţi ca, dintre toate pozele (atributul FOTOGRAFIE_BULETIN) pe care le avem într-o tabelă destinată persoanelor, să le extragem numai pe cele în care, pe obrazul stâng apare o cicatrice. Lucrul acesta este posibil, în prezent, doar cu un software super-specializat.

Date, ore şi intervale calendaristice

În privinţa datelor temporale Joe Celko operează o delimitare între[25]:

- *evenimente*, pentru care există tipurile DATE (zi, lună, an), TIME (oră, minut, secundă), TIMESTAMP (este o combinaţie DATE + TIME + fracţiuni (de ordinul 6) de secundă),
- *intervale* – perioade de timp dintre două evenimente (momente) - tipul INTERVAL (YEAR TO MONTH sau DAY TO HOUR/MINUTE/ SECOND) şi
- *perioade* – sunt intervale cu un moment fix de început, pentru care este necesară o combinaţie de două câmpuri, fie TIMESTAMP - TIMESTAMP, fie TIMESTAMP - INTERVAL.

Fără altă clauză, ora considerată de sistem este UTC (Universal Coordonated Time) sau GMT (Greenwich Mean Time), cum i se spunea anterior. Există însă

---

24 Gestiunea seturilor naţionale de caractere este foarte importantă şi pentru ordonări (Ş urmează după S) sau comparaţii. Primele asemenea opţiuni au fost introduse însă din SQL-92.

25 [Celko 1999]

posibilitatea de a declara şi ora unei alte zone orare (Bucureştiul are ora GMT+2), aşa încât la tipurile de mai sus mai socotiţi două:

- TIME WITH TIME ZONE;
- TIMESTAMP WITH TIME ZONE;

Pentru aceste două tipuri, diferenţa faţă de ora UTC se exprimă ca interval de tip HOUR TO MINUTE cu valori de la -12:59 la +13:00[26].

În privinţa intervalelor de date, acestea pot conţine şi ani şi luni (INTERVAL YEAR TO MONTH), numai ani (INTERVAL YEAR) sau numei luni (INTERVAL MONTH. Dacă mărimea intervalului o cere, se poate specifica numărul de poziţii al anilor (atât pentru INTERVAL YEAR, cât şi INTERVAL YEAR TO MONTH) sau lunilor (numai pentru INTERVAL MONTH). De exemplu, când un interval în care interesează deopotrivă anii şi lunile se întinde pe maxim sute de ani, se recomandă folosirea tipului INTERVAL YEAR(3) TO MONTH. Dacă nu ne-ar interesa decât anii, atunci tipul al fi INTERVAL YEAR(3). Când acelaşi interval trebuie exprimat doar în luni, se foloseşte tipul INTERVAL MONTH(4). În nici un caz nu se poate declara INTERVAL YEAR(2) TO MONTH(3).

Aceeaşi regulă (a specificării numărului de poziţii doar pentru prima dintre mărimile de exprimare a intervalului) se foloseşte şi pentru: INTERVAL DAY(n) TO HOUR, INTERVAL DAY(n) TO MINUTE, INTERVAL DAY(n) TO SECOND, INTERVAL HOUR(n) TO SECOND, INTERVAL MINUTE(n), INTERVAL SECOND(n) etc.

Valori logice

Paradoxal, până în SQL:1999 tabelele nu puteau conţine atribute ale căror valori să fie valorie logice TRUE şi FALSE (plus NULL-ul de rigoare). Şi mai paradoxal, nici acum servere de baze de date din categoria „super grea" (Oracle şi DB2) nu au această „facilitate". Fireşte, lucrurile nu sunt atât de negre, pentru că soluţii ocolitoare există: atribute de tip NUMBER(1) pentru care se declară o restricţie prin care valorile pot fi numai 0 sau 1; atribute de tip CHAR(1) cu singurele valori permise „D" sau "N". Începând cu cel de-al treilea standard, SQL introduce tipul BOOLEAN.

Colecţii

În decursul anilor, au existat multe discuţii pe marginea tipurilor colecţii. Abia în SQL:1999 a fost introdus (doar) tipul ARRAY[27], la care, în SQL:2003 s-a adăugat

---

[26] Logic părea ca intervalul să fie între -11:59 şi 12:00, însă din cauza orei de vară este necesar ca a doua valoare să fie 13:00. Vezi [Melton & Simon 2002], pp.39-40

MULTISET. Tipul ARRAY este, prin comparaţie cu alte limbaje de programare, unul restrictiv, în sensul că elementele unui masiv nu pot fi, la rândul lor, masive, iar colecţia este unidimensională (vector, şi nu matrice).

Exemplu: *INTEGER ARRAY [10]* specifică un tip colecţie cu maximum 10 elemente, fiecare element fiind un număr întreg. Pe de altă parte, numărul maxim de componente specificat (10) nu înseamnă că pentru fiecare variabilă sau atribut de acest tip vor fi automat rezervate toate cele zece poziţii (lucrurile se prezintă oarecum similar raportului VARCHAR/CHAR). Pentru o variabilă sau un atribut definit pe tipul ARRAY de mai sus, numărul de componente al valorii curente reprezintă cardinalitatea curentă, în timp ce 10 este cardinalitatea maximă[28].

În ciuda relativei sărăcii din standardele SQL, fiecare SGBD are propriile tipuri de colecţii, mult mai generoase, după cum vom vedea şi în capitolele din partea a doua a lucrării.

Linii „anonime"

Un alt tip consacrat de SQL:1999 este tipul *tuplu* sau tipul *linie „anonimă"*. Discuţia pare ciudată, întrucât tuplul este o noţiune fundamentală a modelului relaţional. Noutate ţine de faptul că până în SQL:1999 o variabilă sau parametru nu putea fi declarată de tip linie, deşi în multe SGBD-uri lucrul acesta era posibil. Formatul este similar, până la un punct, declarării atributelor unei tabele sau unui tip definit de utilizator:

```
ROW (cnp NUMBER(13), nume VARCHAR(50), prenume VARCHAR(50) )
```

Referinţe

Nu este vorba, după cum probabil v-aţi imaginat (ca mine), de un tip legat de restricţii referenţiale, ci de o trimitere la modelul obiectual, de fapt, la modelul relaţional-obiectual al cărui portdrapel încearcă SQL să fie. În capitolul 19 vom discuta despre tabele obiect în care fiecare linie conţine o instanţă a unei clase (tip) de obiecte. Or, orientarea pe obiecte a consacrat ideea de *identificator unic al obiectului* (OID) – o valoare ce diferenţiază un obiect de oricare altul. Valoarea unică ce diferenţiază o linie-obiect dintr-o asemenea tabelă este denumită valoare-referinţă, iar tipul acesteia tip-referinţă[29].

---

27 [Turker & Gertz 2001], [Melton & Simon 2002], p.42

28 [Melton & Simon 2002], p.43

29 [Melton & Simon 2002], pp.46-47

Locatori

Aplicaţiile complexe care folosesc baze de date în care apar atribute de tip CLOB sau BLOB, cu dimensiuni „mega-litice" sau „giga-litice", ridică importante probleme de performanţă (viteză de lucru), dat fiind imensul transfer dintre platforma client (staţia de lucru) şi serverul BD aflate uneori la distanţă apreciabilă. Tipul locator a fost gândit tocmai pentru dezvoltatorii de aplicaţii şi mai puţin pentru stocarea datelor în bază. De fapt, locatorul este valabil numai pe stratul interfaţă al aplicaţiei, fiind imposibil de folosit la definirea tipului atributelor în tabele[30]. Există trei categorii de locatori, pentru obiecte mari, pentru tipuri-utilizator şi pentru colecţii. Nefiind un tip util stocării datelor în bază, ne oprim aici cu discuţia.

Tipuri definite de utilizator

Un pas imens în evoluţia SQL făcut de SQL:1999 îl reprezintă libertatea utilizatorilor (mă rog, a celor care au suficiente drepturi sau „privilegii") de a defini şi stoca în baza de date a unor tipuri noi, în funcţiile de cerinţele aplicaţiilor. Despre această realizare care consacră modelul relaţional extins, sau relaţional-obiectual, vom discuta în capitolul 19.

3.3. Tabele şi restricţii în SQL

Crearea unei baze de date are o componentă tehnică pronunţată, mai ales în cazul serverelor de baze de date: Oracle, DB2, SQL Server, Informix, PostgreSQL, MySQL etc. Chiar dacă există instrumente grafice care uşurează considerabil această activitate, crearea unei baze de date necesită cunoştinţe de administrare, strict legate de produsul software de gestiune a bazei. În cele ce urmează, vom eluda elementele tehnice/fizice (spaţiile-tabelă, segmentele de rollback, fişierele de date, procesele sistem etc.), lăsându-le în seama administratorilor BD[31]; vom considera baza de date creată din punct de vedere fizic, astfel încât ne interesează numai crearea tabelelor, modificarea structurii lor şi declararea restricţiilor.

30 [Melton & Simon 2002], pp.47

31 Parcurgerea paşilor necesari creării unei baze de date în Oracle este descrisă în [Fotache s.a. 2003]

3.3.1. Crearea tabelelor

Comanda SQL utilizată pentru crearea unei tabele este CREATE TABLE. Precizăm că este vorba de crearea *structurii* tabelei. Popularea cu înregistrări şi actualizarea ulterioară se realizează prin alte comenzi: INSERT, UPDATE, DELETE. Tabelele sunt similare celei din baza de date VÎNZĂRI. Pentru crearea unei tabele şi declararea atributelor acesteia, fără nici o referire la restricţii, comanda CREATE TABLE are, în cazul tabelei JUDETE, următoarea sintaxă:

```
CREATE TABLE judete (
        Jud CHAR(2),
        Judet    VARCHAR(25),
        Regiune VARCHAR (15)
    ) ;
```

Întrucât atributele acestei tabele sunt exclusiv de tip şir de caractere, s-au folosit opţiunile CHAR şi VARCHAR. CHAR este preferat pentru atributul Jud, deoarece indicativul auto al judeţului este format exclusiv din două litere (cu excepţia Bucureştiului). Întrucât denumirile judeţului şi regiunii au lungime variabilă, pentru aceste atribute a fost preferat tipul VARCHAR. Pentru ilustrarea şi a altor două tipuri de date, prezentăm comanda de creare a tabelei FACTURI:

```
CREATE TABLE facturi (
        NrFact NUMERIC(8),
        DataFact DATE,
        CodCl DECIMAL(6),
        Obs VARCHAR(50)
    ) ;
```

NrFact şi CodCl sunt de tip numeric, în timp ce DataFact de tip dată calendaristică. Modelul relaţional nu face referire explicită la această noţiune, însă SGBD-urile permit declararea, pentru fiecare atribut, a unei valori implicite (sau valoarea „din oficiu") – DEFAULT. Iată un exemplu:

```
CREATE TABLE facturi (
        NrFact NUMERIC(8),
        DataFact DATE DEFAULT CURRENT_DATE,
        CodCl DECIMAL(6) DEFAULT 1001,
        Obs VARCHAR(50)
    ) ;
```

Ca urmare a clauzei DEFAULT, în orice linie adăugată în tabela FACTURI, valoarea implicită (dacă nu este specificată în comanda INSERT) a DataFact va fi data sistemului (data curentă), iar CodCl se va iniţializa cu valoarea 1001.

La modul general, când se inserează o linie într-o tabelă (vezi comenzile INSERT din paragraful 3.4), pentru fiecare atribut la care nu se specifică valoarea,

aceasta va fi, automat, cea declarată în clauza DEFAULT. Când clauza DEFAULT lipseşte, iar valoarea atributului nu se specifică în comanda INSERT, atributul va „primi" valoarea NULL. Clauza DEFAULT poate conține, pe lângă literali/constante, funcții sistem de tip dată/timp – CURRENT_DATE, CURRENT_TIME, CURRENT_TIMESTAMP, LOCAL_TIMESTAMP, funcții sistem de genul CURRENT_USER, SESSION_USER, SYSTEM_USER, sau NULL.

Ar mai trebui spus că o tabelă poate fi creată şi pe baza unei interogări, după cum vom vedea la începutul capitolului 13, capitol în care ne vom ocupa şi de crearea tabelelor temporare globale şi locale.

Paragrafele următoare, 3.3.2-3.3.6, ilustrează modul în care pot fi declarate în SQL restricțiile discutate în capitolul precedent.

3.3.2. Restricția de domeniu

Stardardele SQL definesc noțiunea de domeniu, precum şi comanda pentru declararea unui domeniu – CREATE DOMAIN. Ținând seama că există doar câteva regiuni etnografice în țară, putem defini explicit un domeniu care să conțină valorile „autorizate":

```
CREATE DOMAIN dom_regiuni VARCHAR(20)
        CHECK (VALUE IN ('Dobrogea', 'Moldova', 'Muntenia', 'Oltenia', 'Transilvania'))
```

Dorim ca denumirea unui județ să fie introdusă în baza de date pe un şablon unitar: prima literă din fiecare cuvânt al denumirii județului să fie majusculă, iar restul literelor să fie „minuscule"; în plus, denumirea județului nu trebuie să conțină spații la început. Putem crea un domeniu pentru acest tip de valori:

```
CREATE DOMAIN dom_judet VARCHAR(30)
        CHECK (
        VALUE IS NOT NULL AND VALUE = TRIM (LEADING FROM INITCAP(VALUE))
                ) ;
```

La crearea unei tabele – să-i zicem JUDETE_TEST, ultimele două atribute vor fi definite pe aceste domenii:

```
CREATE TABLE judete_test (
        jud CHAR(2),
        judet dom_judet,
```

```
    regiune dom_regiuni
    ) ;
```

Sintaxa de mai sus funcționează în puține SGBD-uri (comenzile prezentate au fost rulate în PostgreSQL). O parte din membrii comitetului de standardizare ANSI/INCTIS au afirmat în repetate rânduri că noțiunea de domeniu va fi eliminată din standardele următoare[32]. Este, probabil, unul dintre motivele pentru care SGBD-uri importante (ex. Oracle, DB2, SQL Server) nu lucrează cu domenii de maniera prezentată în standard.

3.3.3. Restricția de nenulitate

Unele atribute, obligatoriu cele din cheia primară, nu pot prezenta valori nule. Pentru aceasta, la crearea tabelei şi declararea atributului se foloseşte opțiunea NOT NULL:

```
CREATE TABLE facturi (
    NrFact NUMERIC(8) NOT NULL,
    DataFact DATE DEFAULT CURRENT_DATE NOT NULL,
    CodCl DECIMAL(6) DEFAULT 1001 NOT NULL,
    Obs VARCHAR(50)
    ) ;
```

3.3.4. Cheie primară/unicitate

Cheia primară a unei relații este definită prin clauza PRIMARY KEY, plasată fie imediat, după atributul cheie, fie după descrierea ultimului atribut al tabelei. A doua variantă este întrebuințată cu precădere atunci când cheia primară a tabelei este compusă. Pentru atributele de tip cheie candidat se poate folosi clauza UNIQUE care va asigura respectarea unicității valorilor. Iată câteva variante de folosire a clauzelor PRIMARY KEY şi UNIQUE:

```
CREATE TABLE facturi (
    NrFact NUMERIC(8) NOT NULL PRIMARY KEY,
    DataFact DATE DEFAULT CURRENT_DATE NOT NULL,
    CodCl DECIMAL(6) DEFAULT 1001 NOT NULL,
    Obs VARCHAR(50)
```

[32] Vezi, spre exemplu, [Melton & Simon 2002], p.98

```
        ) ;
CREATE TABLE judete (
        Jud CHAR(2) PRIMARY KEY,
        Judet VARCHAR(25) NOT NULL UNIQUE,
        Regiune VARCHAR (15)
        ) ;
CREATE TABLE liniifact (
        NrFact NUMERIC(8) NOT NULL,
        Linie SMALLINT NOT NULL,
        CodPr NUMERIC(6) NOT NULL,
        Cantitate NUMERIC(10) NOT NULL,
        PretUnit NUMBER (12),
        PRIMARY KEY (NrFact, Linie), UNIQUE (NrFact, CodPr)
        ) ;
```

Pentru tabela LINIIFACT cheia primară este combinaţia de atribute (NrFact, Linie); restricţia de unicitate pentru cuplul (NrFact, CodPr) înseamnă că se interzice ca, pe o factură, un produs să apară repetat.

3.3.5. Restricţii referenţiale

Declarea restricţiilor referenţiale se realizează utilizând clauza FOREIGN KEY. Astfel, pentru stabilirea legăturii LINIIFACT-FACTURI comanda de creare are acum formatul:

```
CREATE TABLE liniifact (
        NrFact NUMERIC(8) NOT NULL,
        Linie SMALLINT NOT NULL,
        CodPr NUMERIC(6) NOT NULL,
        Cantitate NUMERIC(10) NOT NULL,
        PretUnit NUMBER (12),
        PRIMARY KEY (NrFact, Linie),
        UNIQUE (NrFact, CodPr),
        FOREIGN KEY NrFact REFERENCES  facturi  (NrFact),
        FOREIGN KEY CodPr REFERENCES  produse(CodPr)
        ) ;
```

Echivalent acestuia, este şi sintaxa în care renunţăm la clauza FOREIGN KEY, păstrând doar REFERENCES plasată în dreptul atributelor cheiei străină:

```
CREATE TABLE liniifact (
        NrFact NUMERIC(8) NOT NULL REFERENCES facturi (NrFact),
        Linie SMALLINT NOT NULL,
        CodPr NUMERIC(6) NOT NULL REFERENCES produse (CodPr),
```

```
    Cantitate NUMERIC(10) NOT NULL, PretUnit NUMBER (12),
    PRIMARY KEY (NrFact, Linie),  UNIQUE (NrFact, CodPr)
    ) ;
```

În SQL se poate specifica modul în care va fi păstrată integritatea bazei de date la ştergerea unei linii-părinte sau modificarea unei chei primare ce prezintă înregistrări-copil. Pentru a interzice ştergerea unor facturi (înregistrări din FACTURI) pentru care există tupluri corespondente în LINIIFACT şi, pe de altă parte, pentru ca la modificarea unui număr de factură (NrFact) în FACTURI să se modifice automat, în cascadă, toate liniile-copil din LINIIFACT, formatul comenzii se schimbă astfel:

```
CREATE TABLE liniifact (
    NrFact NUMERIC(8) NOT NULL,
    Linie SMALLINT NOT NULL,
    CodPr NUMERIC(6) NOT NULL,
    Cantitate NUMERIC(10) NOT NULL,
    PretUnit NUMBER (12),
    PRIMARY KEY (NrFact, Linie),
    UNIQUE (NrFact, CodPr),
    FOREIGN KEY NrFact REFERENCES  facturi (NrFact)
        ON DELETE RESTRICT ON UPDATE CASCADE,
    FOREIGN KEY CodPr REFERENCES  produse (CodPr)
        ON DELETE RESTRICT ON UPDATE CASCADE
    ) ;
```

Există diferențe semnificative între serverele BD în privința definirii acțiunilor ce trebuie întreprinse pentru respectarea integrităților referențiale. Unele, conforme standardului, permit, la declararea restricțiilor referențiale (este de obicei, simultană cu crearea tabelelor), instituirea oricărei reguli din cele patru: ON UPDATE CASCADE, ON UPDATE RESTRICT, ON DELETE CASCADE, ON DELETE RESTRICT.

Altele, precum Oracle sau DB2, permit declarea acțiunilor numai pentru ştergere, ON DELETE CASCADE şi ON DELETE RESTRICT. Pentru aceste servere, modificarea în cascadă a valorilor cheilor primare în înregistrările copil este posibilă doar prin declanşatoare (vezi capitolul 17).

3.3.6. Restricții-utilizator

Restricțiile-utilizator, denumite şi restricții de comportament, sunt implementate de obicei sub forma regulilor de validare la nivel de câmp (*field validation rule*), la nivel de înregistrare (*record validation rule*) sau, eventual, pot fi incluse în declanşatoare (triggere). Cu rezerva că aceste reguli pot avea forme complexe şi întrebuința elemente avansate de SQL şi/sau extensiile procedurale

ale SQL în mediul respectiv, prezentăm în continuare o regulă de validare pentru atributul DataFact în tabela FACTURI şi o alta pentru atributul Linie în LINIIFACT.

```
CREATE TABLE facturi (
        NrFact NUMERIC(8)  nn_facturi_nrfact NOT NULL
                CONSTRAINT pk_facturi PRIMARY KEY,
        DataFact DATE DEFAULT CURRENT_DATE
                CONSTRAINT nn_datafact NOT NULL
                CONSTRAINT ck_datafact CHECK (
                DataFact >= DATE'2007-08-01' AND DataFact <= DATE'2015-12-31'),
        CodCl DECIMAL(6) DEFAULT 1001
        CONSTRAINT nn_facturi_codcl NOT NULL
        CONSTRAINT fk_facturi_clienti REFERENCES CLIENTI(CodCl)
        ON DELETE RESTRICT ON UPDATE CASCADE,
        Obs VARCHAR(50)
        ) ;
CREATE TABLE liniifact (
        NrFact NUMERIC(8) CONSTRAINT nn_liniifact_nrfact NOT NULL,
        Linie SMALLINT CONSTRAINT nn_linie NOT NULL
                CONSTRAINT ck_linie CHECK (Linie > 0),
        CodPr NUMERIC(6) CONSTRAINT nn_liniifact_codpr NOT NULL,
        Cantitate NUMERIC(10) CONSTRAINT nn_cantitate NOT NULL,
        PretUnit NUMERIC (12)
                CONSTRAINT nn_pretunit NOT NULL,
                CONSTRAINT pk_liniifact PRIMARY KEY (NrFact, Linie),
        CONSTRAINT un_liniifact UNIQUE (NrFact, CodPr),
        CONSTRAINT fk_liniifact_facturi FOREIGN KEY NrFact
                REFERENCES  facturi (NrFact)
                        ON DELETE RESTRICT ON UPDATE CASCADE,
        CONSTRAINT fk_liniifact_produse FOREIGN KEY CodPr
                REFERENCES  produse (CodPr)
                        ON DELETE RESTRICT ON UPDATE CASCADE
        ) ;
```

Ca urmare a clauzei CHECK, data unei facturi nu poate avea valori în afara intervalului 1 august 2007 – 31 decembrie 2015, iar atributul Linie trebuie să prezinte valori întregi mai mari ca 0.

Ca un alt element de noutate, fiecărei restricţii (cheie primară, unicitate, regulă la nivel de atribut etc.) i s-a dat un nume, lucru util atunci când, la un eveniment de genul salvare, restaurare, încarcarea BD, vrem sa dezactivăm una sau mai multe

dintre acestea. Pentru a uşura lucrul, se prefixează numele fiecărei restricții cu tipul său:

- pk_ (PRIMARY KEY) pentru cheile primare
- un_ (UNIQUE) pentru cheile alternative
- nn_ (NOT NULL) pentru atributele obligatorii (ce nu pot avea valori nule)
- ck_ (CHECK) pentru reguli de validare la nivel de atribut
- fk_ (FOREIGN KEY) pentru cheile străine.

3.3.7. Modificarea structurii unei tabele

Odată creată o tabelă, schema sa rămâne constantă. Dacă totuşi, ulterior, se doreşte introducerea, modificarea sau ştergerea unor atribute, sau declararea/anularea unor restricții, comanda necesară este ALTER TABLE. Câteva exemple:

- Pentru adăugarea atributului DataNast (data naşterii) în tabela PERSOANE:

```
ALTER TABLE PERSOANE ADD COLUMN DataNast DATE
```

- Ştergerea unui atribut:

```
ALTER TABLE PERSOANE DROP COLUMN DataNast
```

- Modificarea tipului/lungimii unui atribut:

```
ALTER TABLE PERSOANE ALTER COLUMN Nume VARCHAR(21)
```

- Adăugarea/modificarea valorii implicite a unui atribut:

```
ALTER TABLE PERSOANE ALTER COLUMN Sex SET DEFAULT 'F'
```

- Anularea unei valori implicite:

```
ALTER TABLE PERSOANE ALTER COLUMN Sex DROP DEFAULT
```

- Instituirea restricției de nenulitate:

```
ALTER TABLE PERSOANE ADD CONSTRAINT nn_sex Sex IS NOT NULL
```

- Invers, acceptarea valorilor NULL:

```
ALTER TABLE PERSOANE DROP CONSTRAINT nn_sex
```

- Adăugarea unei reguli de validare la nivel de atribut:

```
ALTER TABLE PERSOANE ADD CONSTRAINT ck_sex CHECK (sex IN ('B', 'F'))
```

- Ştergerea unei reguli de validare lal nivel de atribut:

```
ALTER TABLE PERSOANE DROP CONSTRAINT ck_sex
```

Toate restricțiile: cheie primară - PRIMARY KEY, unicitate - UNIQUE, referențială - FOREIGN KEY, de comportament - CHECK pot fi declarate ulterior creării tabelei şi, bineînțeles, anulate la un moment dat. Spre exemplu, formatul general al comenzii pentru dezactivarea cheii primare este:

ALTER TABLE persoane DROP PRIMARY KEY.

3.3.8. Ştergerea tabelelor

Tabelele pot fi şterse prin comanda DROP TABLE. Dacă, însă, tabela este una părinte (adică are copii), ştergerea trebuie făcută împreună cu toți copiii (DELETE CASCADE), altminteri SGBD-ul se va împotrivi (va afişa un mesaj de încălcare a restricției referențiale). De multe ori, scripturile de creare a tabelelor unei baze de date (succesiunea comenzilor CREATE TABLE) încep cu DROP TABLE, tocmai pentru a evita erorile datorate încercării de a crea o tabelă care există deja în schema respectivă. La prima execuție (sau prima lansare de după o execuție incompletă) a scriptului, este posibil ca o comandă DROP TABLE să genereze o eroare (deoarece se încearcă ştergerea unei tabele inexistente). Lucrul acesta poate fi evitat elegant în PostgreSQL prin folosirea sintaxei *DROP IF EXISTS judete*. În DB2, SQL Server şi Oracle trebuie consultat dicționarul de date pentru a vedea dacă tabela există, formatul general fiind ceva mai complicat.

3.3.9. Scripturi de creare a tabelelor în DB2

Dialectul SQL din DB2 este destul de aproape de standardele SQL (unii ar spune că şi reciproca este valabilă). Tipurile de date predefinite "clasice" se regăsesc întocmai în DB2. Iată care ar fi scriptul de crearea a celor patru tabele din BD TRIAJ (listing 3.1):

Listing 3.1 Script DB2 de creare a tabelelor BD TRIAJ

```
DROP TABLE triaj ;
DROP TABLE pacienti ;
DROP TABLE garzi ;
DROP TABLE doctori ;

CREATE TABLE doctori (
    iddoctor SMALLINT NOT NULL
        GENERATED ALWAYS AS IDENTITY (START WITH 1
                INCREMENT BY 1 MINVALUE 1 NO MAXVALUE)
        CONSTRAINT pk_doctori PRIMARY KEY,
    numedoctor VARCHAR(50) NOT NULL,
    specialitate VARCHAR(40),
    datanasterii DATE
    );

CREATE TABLE garzi (
    iddoctor SMALLINT NOT NULL REFERENCES doctori (iddoctor),
    inceput_garda TIMESTAMP DEFAULT CURRENT_TIMESTAMP NOT NULL,
    sfirsit_garda TIMESTAMP ,
    CONSTRAINT pk_garzi PRIMARY KEY (iddoctor, inceput_garda)
    ) ;
```

```
CREATE TABLE pacienti (
    idpacient INTEGER NOT NULL GENERATED ALWAYS AS IDENTITY
                (START WITH 1 INCREMENT BY 1 MINVALUE 1 NO MAXVALUE)
        CONSTRAINT pk_pacienti PRIMARY KEY,
    numepacient VARCHAR(60),
    CNP NUMERIC(13),
    adresa VARCHAR(100),
    loc VARCHAR(30),
    judet CHAR(2),
    tara VARCHAR(30) DEFAULT 'Romania',
    serie_nr_act_identitate VARCHAR(20)
    );

CREATE TABLE triaj (
    idexaminare BIGINT NOT NULL GENERATED ALWAYS AS IDENTITY
        (START WITH 1 INCREMENT BY 1 MINVALUE 1 NO MAXVALUE),
    dataora_examinare TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    idpacient NUMERIC(10) NOT NULL CONSTRAINT fk_internari_pacienti
        REFERENCES pacienti (idpacient),
    simptome VARCHAR(500),
    tratament_imediat VARCHAR(500),
    sectie_destinatie VARCHAR(30)
     );
```

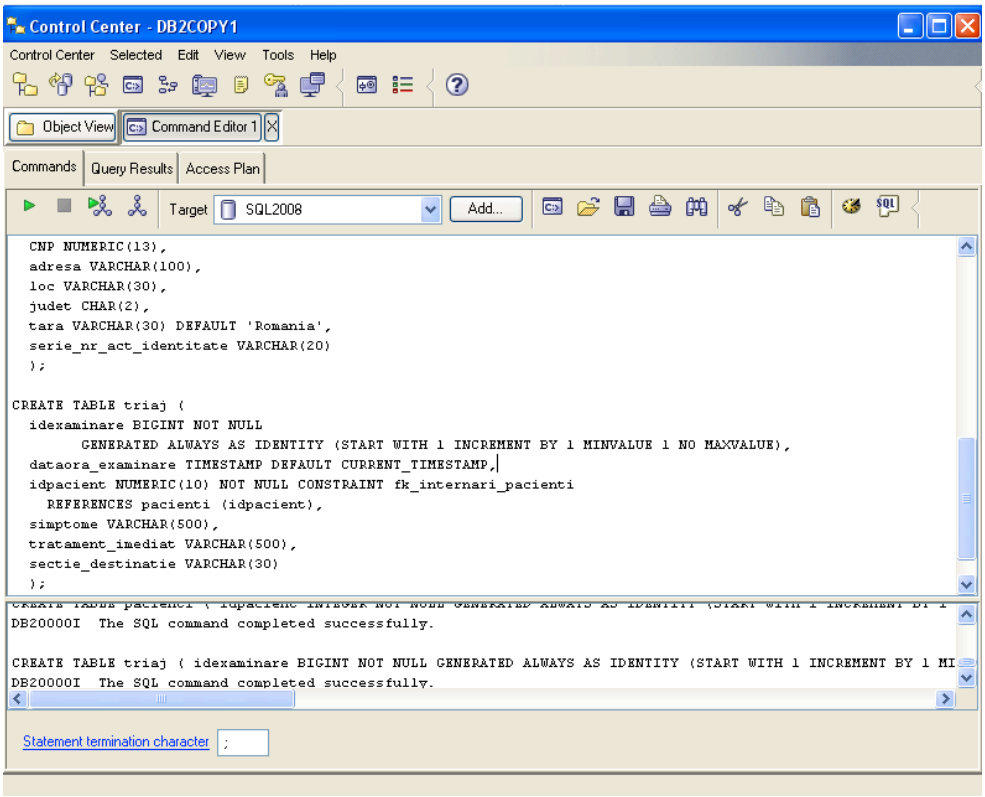


Figura 3.2. Lansarea în execuţie în DB2 a scriptului (comenzilor) de creare a tabelelor BD TRIAJ

Lansarea în execuție a acestui script se face din Control Center, folosind, din meniu, opțiunea *Command Editor* – vezi figura 3.2.

Pentru atributul IdDoctor din tabela DOCTORI am ales tipul SMALLINT ale cărui valori minimă şi maximă sunt (în DB2) -32768 şi 32767. În schimb, pentru IdPacient tipul ales a fost INTEGER (pentru, că, în decursul anilor/deceniilor, numărul pacienților poate fi de ordinul sutelor de mii, poate chiar milioanelor (deşi e cam exagerat!)), în timp ce pentru IdExaminare (tabela TRIAJ) tipul ales a fost (tot exagerat) BIGINT. Valorile minime şi maxime ale tipurilor INTEGER şi BIGINT sunt în DB2 -2147483648 şi 2147483647, respectiv (aproximativ) $\pm 9{,}2 * 10^{18}$.

IdDoctor, IdPacient şi IdExaminare sunt chei primare în tabelele DOCTORI, PACIENȚI şi TRIAJ. Neavând o semnificație anume, valorile lor pot fi generate automat (1, 2, 3,…) folosind opțiunea GENERATES ALWAYS AS IDENTITY…, ceea ce am şi făcut. Vom vedea, peste un număr de pagini, că aceste coloane autogenerate au un anumit regim la popularea şi modificarea înregistrărilor din tabelele în care apar.

În privința scriptului de creare a tabelelor bazei de date VÎNZĂRI (listingul 3.2), vom încerca, la fel ca şi în viitoarele scripturi dedicare celorlalte trei servere BD, să implementăm cât mai multe dintre restricțiile definite în paragraful 2.3.2, subliniind eventualele particularități. Spre exemplu, în DB2 nu există funcția INITCAP, şi nici o alta echivalentă care să ne permită verificarea majusculei fiecărui cuvânt dintr-un şir de caractere. De asemenea, la regulile de validare pentru atributele DataFact din tabela FACTURI, DataInc şi DataDoc din tabela INCASFACT, constantele de tip dată calendaristică se scriu "direct" (*datafact* >= *'2007-01-01'*), şi nu conform standardului SQL (*datafact* >= *DATE '2007-01-01'*). Se mai poate folosi însă şi formatul *DATE('2007-01-01')*.

Listing 3.2 Script DB2 de creare a tabelelor BD VÎNZĂRI

```
DROP TABLE incasfact  ;
DROP TABLE incasari ;
DROP TABLE liniifact ;
DROP TABLE facturi ;
DROP TABLE produse ;
DROP TABLE persclienti ;
DROP TABLE persoane ;
DROP TABLE clienti ;
DROP TABLE coduri_postale ;
DROP TABLE judete ;

CREATE TABLE judete (
    jud CHAR(2) NOT NULL CONSTRAINT pk_judete PRIMARY KEY
        CONSTRAINT ck_jud CHECK (jud=LTRIM(UPPER(jud))),
    judet VARCHAR(25)
        CONSTRAINT un_judet UNIQUE CONSTRAINT nn_judet NOT NULL,
    regiune VARCHAR(15) DEFAULT 'Moldova'
        CONSTRAINT nn_regiune NOT NULL
        CONSTRAINT ck_regiune CHECK (regiune IN ('Banat', 'Transilvania',
```

```
                    'Dobrogea',  'Oltenia', 'Muntenia', 'Moldova'))
    ) ;

CREATE TABLE coduri_postale (
    codpost CHAR(6) NOT NULL
        CONSTRAINT pk_coduri_post PRIMARY KEY
        CONSTRAINT ck_codpost CHECK (codpost=LTRIM(codpost)),
    loc VARCHAR(25) CONSTRAINT nn_loc NOT NULL,
    jud CHAR(2) DEFAULT 'IS'
                    CONSTRAINT fk_coduri_post_jud REFERENCES judete(jud)
    ) ;

CREATE TABLE clienti (
    codcl SMALLINT NOT NULL
        GENERATED ALWAYS AS IDENTITY (START WITH 1001
                INCREMENT BY 1 MINVALUE 1 NO MAXVALUE)
        CONSTRAINT pk_clienti PRIMARY KEY
        CONSTRAINT ck_codcl CHECK (codcl > 1000),
    dencl VARCHAR(30) NOT NULL
                    CONSTRAINT ck_dencl CHECK (SUBSTR(dencl,1,1) =
                    UPPER(SUBSTR(dencl,1,1))),
    codfiscal CHAR(9) NOT NULL
        CONSTRAINT un_codfiscal UNIQUE
        CONSTRAINT ck_codfiscal CHECK (SUBSTR(codfiscal,1,1) =
                UPPER(SUBSTR(codfiscal,1,1))),
    adresa VARCHAR(40) CONSTRAINT ck_adresa_clienti
CHECK (SUBSTR(adresa,1,1) = UPPER(SUBSTR(adresa,1,1))),
    codpost CHAR(6) NOT NULL
        CONSTRAINT fk_clienti_cp REFERENCES coduri_postale(codpost),
    telefon VARCHAR(10)
    ) ;

CREATE TABLE persoane (
    cnp CHAR(14) NOT NULL
        CONSTRAINT pk_persoane PRIMARY KEY
        CONSTRAINT ck_cnp CHECK (cnp=LTRIM(UPPER(cnp))),
    nume VARCHAR(20) NOT NULL,
    prenume VARCHAR(20) NOT NULL,
    adresa VARCHAR(50)
        CONSTRAINT ck_adresa_persoane CHECK (SUBSTR(adresa,1,1) =
        UPPER(SUBSTR(adresa,1,1))),
    sex CHAR(1) DEFAULT 'B' NOT NULL
        CONSTRAINT ck_sex CHECK (sex IN ('F','B')),
    codpost CHAR(6) NOT NULL
        CONSTRAINT fk_persoane_cp REFERENCES coduri_postale(codpost),
    telacasa VARCHAR(10),
    telbirou VARCHAR(10),
    telmobil VARCHAR(10),
    email VARCHAR(30)
    ) ;

CREATE TABLE persclienti (
    cnp CHAR(14) NOT NULL
        CONSTRAINT fk_persclienti_persoane REFERENCES persoane(cnp),
    codcl SMALLINT  NOT NULL
        CONSTRAINT fk_persclienti_clienti REFERENCES clienti(codcl),
    functie VARCHAR(25) NOT NULL
        CONSTRAINT ck_functie CHECK (SUBSTR(functie,1,1) =
                UPPER(SUBSTR(functie,1,1))),
```

```
    CONSTRAINT pk_persclienti PRIMARY KEY (cnp, codcl, functie)
    ) ;

CREATE TABLE produse (
    codpr SMALLINT NOT NULL
        GENERATED ALWAYS AS IDENTITY (START WITH 1
                INCREMENT BY 1 MINVALUE 1 NO MAXVALUE)
        CONSTRAINT pk_produse PRIMARY KEY
        CONSTRAINT ck_codpr CHECK (codpr > 0),
    denpr VARCHAR(30) NOT NULL CONSTRAINT ck_denpr
        CHECK (SUBSTR(denpr,1,1) = UPPER(SUBSTR(denpr,1,1))),
    um VARCHAR(10) NOT NULL,
    grupa VARCHAR(15) CONSTRAINT ck_produse_grupa
        CHECK (SUBSTR(grupa,1,1) = UPPER(SUBSTR(grupa,1,1))),
    procTVA NUMERIC(2,2) DEFAULT .19 NOT NULL
    ) ;

CREATE TABLE facturi (
    nrfact INTEGER NOT NULL
                CONSTRAINT pk_facturi PRIMARY KEY,
    datafact DATE DEFAULT CURRENT_DATE NOT NULL,
                CONSTRAINT ck_datafact CHECK (datafact >= '2007-01-01'
                        AND datafact <= '2015-12-31'),
    codcl SMALLINT NOT NULL
                CONSTRAINT fk_facturi_clienti REFERENCES clienti(codcl) ,
    obs VARCHAR(50)
    ) ;

CREATE TABLE liniifact (
    nrfact INTEGER NOT NULL
        CONSTRAINT fk_liniifact_facturi REFERENCES facturi(nrfact),
    linie SMALLINT NOT NULL
        CONSTRAINT ck_linie CHECK (linie > 0),
    codpr SMALLINT NOT NULL
        CONSTRAINT fk_liniifact_produse REFERENCES produse(codpr),
    cantitate NUMERIC(8) NOT NULL,
    pretunit NUMERIC (9,2) NOT NULL,
    CONSTRAINT pk_liniifact PRIMARY KEY (nrfact,linie),
    CONSTRAINT un_liniifact UNIQUE (nrfact,codpr)
    ) ;

CREATE TABLE incasari (
    codinc BIGINT NOT NULL
        GENERATED ALWAYS AS IDENTITY (START WITH 1
                INCREMENT BY 1 MINVALUE 1 NO MAXVALUE)
        CONSTRAINT pk_incasari PRIMARY KEY,
    datainc DATE DEFAULT CURRENT_DATE NOT NULL
        CONSTRAINT ck_datainc CHECK (datainc >= '2007-01-01' AND
                datainc <= '2015-12-31'),
    coddoc CHAR(4) NOT NULL
        CONSTRAINT ck_coddoc CHECK(coddoc=UPPER(LTRIM(coddoc))),
    nrdoc VARCHAR(16) NOT NULL,
    datadoc DATE DEFAULT CURRENT_DATE  NOT NULL
        CONSTRAINT ck_datadoc CHECK (datadoc >= '2007-01-01' AND
                datadoc <= '2010-12-31')
    ) ;

CREATE TABLE incasfact (
    codinc BIGINT NOT NULL
```

```
        CONSTRAINT fk_incasfact_incasari REFERENCES incasari(codinc) ,
    nrfact INTEGER NOT NULL
        CONSTRAINT fk_incasfact_facturi REFERENCES facturi(nrfact),
    transa NUMERIC(11,2) NOT NULL,
    CONSTRAINT pk_incasfact PRIMARY KEY (codinc, nrfact)
    ) ;

ALTER TABLE incasari ADD CONSTRAINT ck_date_incdoc
        CHECK (DataInc >= DataDoc) ;
```

Cheile primare CodCl (din CLIENȚI), CodPr (PRODUSE) şi CodInc (ÎNCASĂRI) au fost declarate atribute auto-generate. Valorile de start sunt: 1001 pentu CodCl, 1 pentru CodPr şi pentru CodInc. În schimb, NrFact (FACTURI) nu, întrucât firma poate avea anumite intervale de valori pe care le poate folosi pentru numerotarea facturilor, aceste intervale nefiind consecutive (ex. Compania numerotază facturile de la 2345678 la 3456789, apoi, de la 4567890 la 5800400 s.a.m.d. Chiar dacă valorile generate pentru NrFact trebuie să fie consecutive, ele nu pot fi autogenerate.

Ultima comandă din script este ALTER TABLE prin care am vrut să demonstrăm că restricțiile pot fi declarate şi ulterior creării tabelei, în cazul nostru fiind vorba de regula la nivel de înregistrare potrivit căreia, în tabela ÎNCASĂRI, data documentului de plată trebuie să fie mai mică sau cel mult egală cu data încasării (data intrării banilor în cont/casierie).

3.3.10. Scripturi de creare a tabelelor în Oracle

Şi Oracle este unul dintre servere BD al căror dialect SQL este foarte apropiat de standard. Ca diferențe de standard şi celelalte servere BD amintim, pentru moment:

- tipul DATE este în Oracle echivalent tipului TIMESTAMP (minus zone orare), întrucât datele de acest tip conțin, pe lângă zi-lună-an şi ora, minutul, secunda şi fracțiuni de secundă; cu toate acestea, noi vom folosi tipul TIMESTAMP, ca în standard;
- tradiția Oracle a consacrat tipul VARCHAR2 în loc de VARCHAR, deşi explicațiile acestei substituiri sunt mai degrabă obscure.
- în Oracle există funcția INITCAP care converteşte prima literă a fiecărui cuvânt dintr-un şir de caractere în majusculă.

Listingul 3.3 conține scriptul de creare a celor patru tabele grupate în BD TRIAJ. Spre deosebire de celelate trei servere BD, Oracle nu prezintă tipurile SMALLINT, INTEGER sau BIGINT. De asemenea, pentru coloanele de tip cheie surogat (IdDoctor, IdPacient, IdExaminare, CodCl, CodPr, CodInc) singurul mecanism care să "automatizeze" valorile acestora este cel bazat pe secvențe şi declanşatoare, mecanism pe care îl vom prezenta abia în cap 17.

Listing 3.3 Script Oracle de creare a tabelelor BD TRIAJ

```
DROP TABLE triaj ;
```

```
DROP TABLE pacienti ;
DROP TABLE garzi ;
DROP TABLE doctori ;

CREATE TABLE doctori (
    iddoctor NUMBER(8) CONSTRAINT pk_doctori PRIMARY KEY,
    numedoctor VARCHAR2(50) NOT NULL,
    specialitate VARCHAR2(40) NOT NULL,
    datanasterii DATE
    );

CREATE TABLE garzi (
    iddoctor NUMBER(8) NOT NULL REFERENCES doctori (iddoctor),
    inceput_garda TIMESTAMP DEFAULT CURRENT_TIMESTAMP
        NOT NULL,
    sfirsit_garda TIMESTAMP,
        CONSTRAINT pk_garzi PRIMARY KEY (iddoctor, inceput_garda)
    ) ;

CREATE TABLE pacienti (
    idpacient NUMBER(10) CONSTRAINT pk_pacienti PRIMARY KEY,
    numepacient VARCHAR2(60),
    CNP NUMBER(13),
    adresa VARCHAR2(100),
    loc VARCHAR2(30),
    judet CHAR(2),
    tara VARCHAR2(30) DEFAULT 'Romania',
    serie_nr_act_identitate VARCHAR2(20)
    );

CREATE TABLE triaj (
    idexaminare NUMBER(12)
        CONSTRAINT pk_internari PRIMARY KEY,
    dataora_examinare TIMESTAMP DEFAULT CURRENT_TIMESTAMP
        NOT NULL,
    idpacient NUMBER(10) NOT NULL
        CONSTRAINT fk_internari_pacienti
                REFERENCES pacienti (idpacient),
    simptome VARCHAR2(500) NOT NULL,
    tratament_imediat VARCHAR2(500),
    sectie_destinatie VARCHAR2(30)
    );
```

Lansarea în execuţie a acestor comenzi folosind SQL Developer este ilustrată în figura 3.3. Neavând la dispoziţia tipurile SMALLINT, INTEGER şi BIGINT, dimensiunea atributelor numerice se stabileşte prin cifra/cifrele argument ale tipului

NUMERIC. Mie mi se pare un pic ciudat ca un server cu atâta forță să nu pună la dispoziție aceste trei tipuri, însă, să recunoaștem, deranjul nu este chiar insuportabil[33].

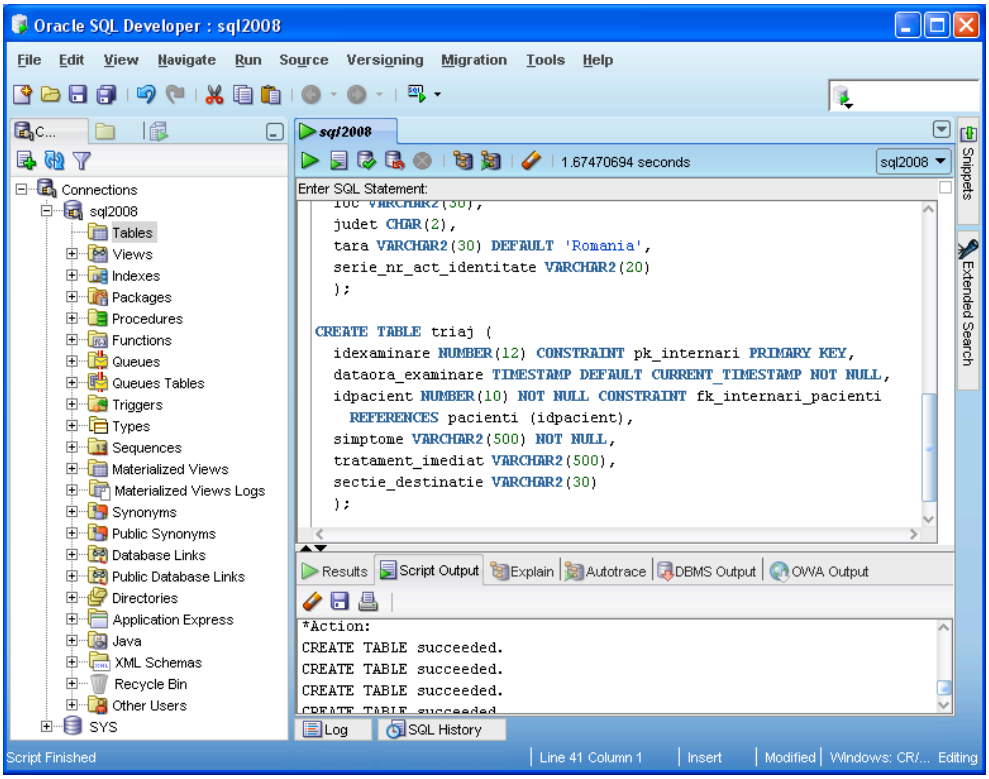


Figura 3.3. Lansarea în execuție în Oracle (SQL Developer) a scriptului (comenzilor) de creare a tabelelor BD TRIAJ

Listingul 3.4 grupează comenzile pentru crearea tabelelor BD VÎNZĂRI. Spre deosebire de varianta DB2, în Oracle în comenzile DROP TABLE, am folosit opțiunea CASCADE CONSTRAINTS care asigură ştergerea unei tabele, chiar dacă mai există în schema BD restricții (cum ar fi cele referențiale) în care tabela respectivă este implicată. De asemenea, clauza DEFAULT a atributului DataDoc

[33] În Oracle există tipuri precum INTEGER, BINARY_INTEGER sau PLS_INTEGER dar numai în extensia procedurală a SQL care este limbajul PL/SQL.

din tabela INCASARI este expresia CURRENT_DATE – 7[34], expresia neacceptată de DB2.

Listing 3.4 Script Oracle de creare a tabelelor BD VÎNZĂRI

```
DROP TABLE incasfact CASCADE CONSTRAINTS;
DROP TABLE incasari CASCADE CONSTRAINTS;
DROP TABLE liniifact CASCADE CONSTRAINTS;
DROP TABLE facturi CASCADE CONSTRAINTS;
DROP TABLE produse CASCADE CONSTRAINTS;
DROP TABLE persclienti CASCADE CONSTRAINTS;
DROP TABLE persoane CASCADE CONSTRAINTS;
DROP TABLE clienti CASCADE CONSTRAINTS;
DROP TABLE coduri_postale CASCADE CONSTRAINTS;
DROP TABLE judete CASCADE CONSTRAINTS;

CREATE TABLE judete (
    jud CHAR(2)
                CONSTRAINT pk_judete PRIMARY KEY
                CONSTRAINT ck_jud CHECK (jud=LTRIM(UPPER(jud))),
    judet VARCHAR2(25)
                CONSTRAINT un_judet UNIQUE
                CONSTRAINT nn_judet NOT NULL
                CONSTRAINT ck_judet CHECK (judet=LTRIM(INITCAP(judet))),
    regiune VARCHAR2(15)
                DEFAULT 'Moldova' CONSTRAINT nn_regiune NOT NULL
                CONSTRAINT ck_regiune CHECK (regiune IN ('Banat', 'Transilvania',
                 'Dobrogea', 'Oltenia', 'Muntenia', 'Moldova'))
    ) ;

CREATE TABLE coduri_postale (
    codpost CHAR(6)
                CONSTRAINT pk_coduri_post PRIMARY KEY
                CONSTRAINT ck_codpost CHECK (codpost=LTRIM(codpost)),
    loc VARCHAR2(25)
                CONSTRAINT nn_loc NOT NULL
                CONSTRAINT ck_loc CHECK (loc=LTRIM(INITCAP(loc))),
    jud CHAR(2) DEFAULT 'IS'
                CONSTRAINT fk_coduri_post_jud REFERENCES judete(jud)
    ) ;

CREATE TABLE clienti (
    codcl NUMERIC(6)
                CONSTRAINT pk_clienti PRIMARY KEY
                CONSTRAINT ck_codcl CHECK (codcl > 1000),
```

[34] Aceasta ar însemna că, în medie, de la data în care clienţii întocmesc documentele de plată până în ziua în care intră banii în contul firmei noastre (şi se operează în baza de date) trec şapte zile, cifră care este mai mare decât în realitate.

```
    dencl VARCHAR2(30) NOT NULL
        CONSTRAINT ck_dencl CHECK (SUBSTR(dencl,1,1) =
                UPPER(SUBSTR(dencl,1,1))),
    codfiscal CHAR(9) NOT NULL
        CONSTRAINT un_codfiscal UNIQUE
        CONSTRAINT ck_codfiscal CHECK (SUBSTR(codfiscal,1,1) =
                UPPER(SUBSTR(codfiscal,1,1))),
    adresa VARCHAR2(40)
        CONSTRAINT ck_adresa_clienti CHECK (SUBSTR(adresa,1,1) =
                UPPER(SUBSTR(adresa,1,1))),
    codpost CHAR(6) NOT NULL
        CONSTRAINT fk_clienti_cp REFERENCES coduri_postale(codpost),
    telefon VARCHAR2(10)
    ) ;

CREATE TABLE persoane (
    cnp CHAR(14)
        CONSTRAINT pk_persoane PRIMARY KEY
        CONSTRAINT ck_cnp CHECK (cnp=LTRIM(UPPER(cnp))),
    nume VARCHAR2(20) NOT NULL
        CONSTRAINT ck_nume CHECK (nume=LTRIM(INITCAP(nume))),
    prenume VARCHAR2(20) NOT NULL
        CONSTRAINT ck_prenume CHECK (prenume=LTRIM(INITCAP(prenume))),
    adresa VARCHAR2(50)
        CONSTRAINT ck_adresa_persoane CHECK (SUBSTR(adresa,1,1) =
                UPPER(SUBSTR(adresa,1,1))),
    sex CHAR(1) DEFAULT 'B' NOT NULL
        CONSTRAINT ck_sex CHECK (sex IN ('F','B')),
    codpost CHAR(6) NOT NULL
        CONSTRAINT fk_persoane_cp REFERENCES coduri_postale(codpost),
    telacasa VARCHAR(10),
    telbirou VARCHAR(10),
    telmobil VARCHAR(10),
    email VARCHAR2(30)
    ) ;

CREATE TABLE persclienti (
    cnp CHAR(14) NOT NULL
        CONSTRAINT fk_persclienti_persoane REFERENCES persoane(cnp),
    codcl NUMERIC(6) NOT NULL
        CONSTRAINT fk_persclienti_clienti REFERENCES clienti(codcl),
    functie VARCHAR2(25) NOT NULL
        CONSTRAINT ck_functie CHECK (SUBSTR(functie,1,1) =
                UPPER(SUBSTR(functie,1,1))),
        CONSTRAINT pk_persclienti PRIMARY KEY (cnp, codcl, functie)
    ) ;

CREATE TABLE produse (
    codpr NUMERIC(6)
        CONSTRAINT pk_produse PRIMARY KEY
        CONSTRAINT ck_codpr CHECK (codpr > 0),
    denpr VARCHAR2(30) NOT NULL CONSTRAINT ck_denpr
        CHECK (SUBSTR(denpr,1,1) = UPPER(SUBSTR(denpr,1,1))),
    um VARCHAR2(10) NOT NULL,
    grupa VARCHAR2(15) CONSTRAINT ck_produse_grupa
        CHECK (SUBSTR(grupa,1,1) = UPPER(SUBSTR(grupa,1,1))),
    procTVA NUMERIC(2,2) DEFAULT .19 NOT NULL
    ) ;
```

```
CREATE TABLE facturi (
    nrfact NUMERIC(8)
        CONSTRAINT pk_facturi PRIMARY KEY,
    datafact DATE DEFAULT CURRENT_DATE NOT NULL,
        CONSTRAINT ck_datafact CHECK (datafact >= DATE'2007-01-01'
                AND datafact <= DATE'2015-12-31'),
    codcl NUMERIC(6) NOT NULL
        CONSTRAINT fk_facturi_clienti REFERENCES clienti(codcl),
    obs VARCHAR2(50)
    ) ;

CREATE TABLE liniifact (
    nrfact NUMERIC(8)
        CONSTRAINT fk_liniifact_facturi REFERENCES facturi(nrfact),
    linie NUMERIC(2) NOT NULL
        CONSTRAINT ck_linie CHECK (linie > 0),
    codpr NUMERIC(6) NOT NULL
        CONSTRAINT fk_liniifact_produse REFERENCES produse(codpr),
    cantitate NUMERIC(8) NOT NULL,
    pretunit NUMERIC (9,2) NOT NULL,
        CONSTRAINT pk_liniifact PRIMARY KEY (nrfact,linie),
        CONSTRAINT un_liniifact UNIQUE (nrfact,codpr)
    ) ;

CREATE TABLE incasari (
    codinc NUMERIC(8)
        CONSTRAINT pk_incasari PRIMARY KEY,
    datainc DATE DEFAULT CURRENT_DATE NOT NULL
        CONSTRAINT ck_datainc CHECK (datainc BETWEEN
                DATE'2007-01-01' AND DATE'2015-12-31'),
    coddoc CHAR(4) NOT NULL
        CONSTRAINT ck_coddoc CHECK(coddoc=UPPER(LTRIM(coddoc))),
    nrdoc VARCHAR(16) NOT NULL,
    datadoc DATE DEFAULT CURRENT_DATE - 7 NOT NULL
        CONSTRAINT ck_datadoc CHECK (datadoc BETWEEN
                DATE'2007-01-01' AND DATE'2015-12-31')
    ) ;

CREATE TABLE incasfact (
    codinc NUMERIC(8) CONSTRAINT fk_incasfact_incasari REFERENCES incasari(codinc),
    nrfact NUMERIC(8) CONSTRAINT fk_incasfact_facturi REFERENCES facturi(nrfact),
    transa NUMERIC(11,2) NOT NULL,
        CONSTRAINT pk_incasfact PRIMARY KEY (codinc, nrfact)
    ) ;

ALTER TABLE incasari ADD CONSTRAINT ck_date_incdoc CHECK (DataInc >= DataDoc) ;
```

3.3.11. Scripturi de creare a tabelelor în PostgreSQL

Chiar dacă este un server open-source, PostgreSQL este înzestrat cu un dialect SQL redutabil. PostgreSQL gestionează şi tipurile SMALLINT, INTEGER şi BIGINT, ale căror valori minime şi maxime sunt similare celor din DB2. Pentru cheile surogat, există două tipuri, SERIAL (care generează valori de la 1 la 2147483647) şi BIGSERIAL (a cărui valoare maxim generată este cea a tipului BIGINT).

Listing 3.5. Script PostgreSQL de creare a tabelelor BD TRIAJ

```
DROP TABLE IF EXISTS triaj ;
DROP TABLE IF EXISTS pacienti ;
DROP TABLE IF EXISTS garzi ;
DROP TABLE IF EXISTS doctori ;

CREATE TABLE doctori (
    iddoctor SERIAL CONSTRAINT pk_doctori PRIMARY KEY,
    numedoctor VARCHAR(50) NOT NULL,
    specialitate VARCHAR(40) NOT NULL,
    datanasterii DATE
    );

CREATE TABLE garzi (
    iddoctor SMALLINT  NOT NULL
        REFERENCES doctori (iddoctor) ON UPDATE CASCADE,
    inceput_garda TIMESTAMP DEFAULT CURRENT_TIMESTAMP NOT NULL,
    sfirsit_garda TIMESTAMP,
    CONSTRAINT pk_garzi PRIMARY KEY (iddoctor, inceput_garda)
    ) ;

CREATE TABLE pacienti (
    idpacient SERIAL CONSTRAINT pk_pacienti PRIMARY KEY,
    numepacient VARCHAR(60),
    CNP NUMERIC(13),
    adresa VARCHAR(100),
    loc VARCHAR(30),
    judet CHAR(2),
    tara VARCHAR(30) DEFAULT 'Romania',
    serie_nr_act_identitate VARCHAR(20)
    );

CREATE TABLE triaj (
    idexaminare BIGSERIAL CONSTRAINT pk_internari PRIMARY KEY,
    dataora_examinare TIMESTAMP DEFAULT CURRENT_TIMESTAMP NOT NULL,
    idpacient INTEGER NOT NULL
        CONSTRAINT fk_internari_pacienti
                REFERENCES pacienti (idpacient) ON UPDATE CASCADE,
    simptome VARCHAR(500) NOT NULL,
    tratament_imediat VARCHAR(500),
    sectie_destinatie VARCHAR(30)
    );
```

Față de variantele din DB2 şi Oracle, în listingul 3.5 am folosit opțiunea IF EXISTS a comenzii DROP TABLE, pentru a preveni afişarea mesajelor de eroare la prima lansare în execuție, atunci când cele patru table nu există, iar comenzile DROP nu au ce şterge. Figura 3.4 surprinde momentul lansării în execuție a scriptului în "clientul" pgAdmin.

Interesant este şi că, dacă în tabela DOCTORI tipul atributului cheie primară (IdDoctor) este SERIAL, acelaşi atribut a fost declarat în tabela copil (GĂRZI) ca fiind de tip SMALLINT. În schimb, IdPacient a fost declarat de tip SERIAL în PACIENȚI, dar de tip INTEGER în tabela TRIAJ. Altminteri, nu sunt mari diferențe față de listingul Oracle (3.3).

Nici listingul următor – 3.6 – nu este mult diferit față de 3.4, dacă facem abstracție de atributele cu valori autogenerate. Diferența principală față de scripturile echivalente DB2 şi Oracle ține de posibilitatea folosirii clauzei ON UPDATE CASCADE la definirea restricțiilor referențiale, opțiune, paradoxal, inexistentă în cele mai pretențioase două servere BD.

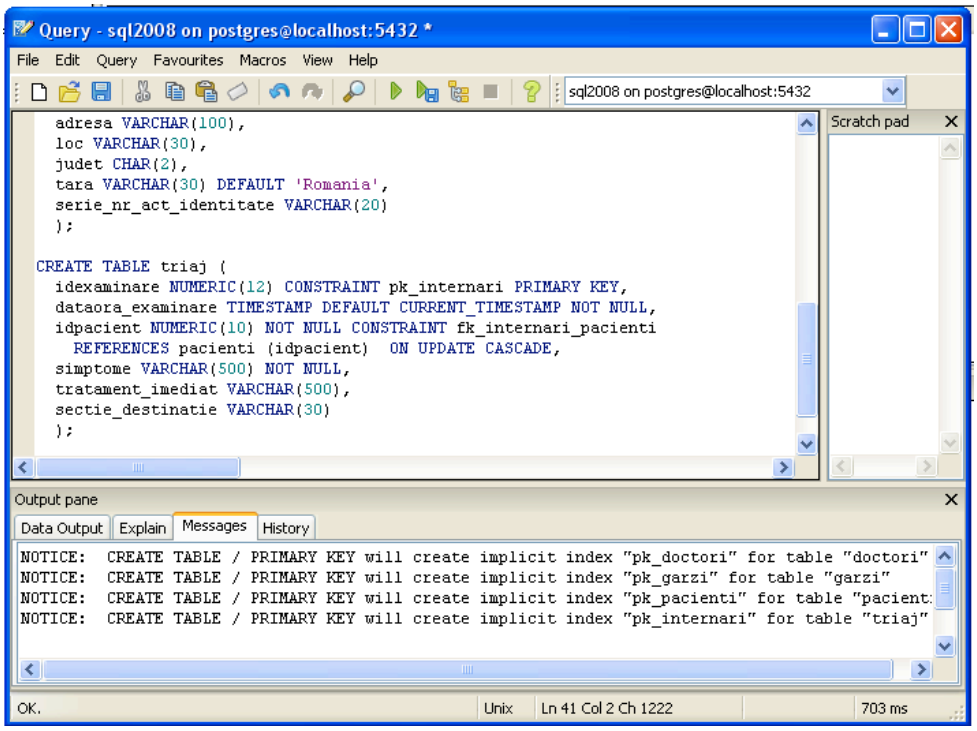


Figura 3.4. Lansarea în execuţie în PostgreSQL (pgAdmin) a scriptului (comenzilor) de creare a tabelelor BD TRIAJ

Ca şi în Oracle, în PostgreSQL o constantă de tip dată calendaristică (vezi tabelele FACTURI şi INCASARI) se pot scrie atât în formatul DATE'an-zi-lună', cât şi cu ajutorul funcției TO_DATE[35]. În plus, similar DB2, constantele de tip dată

[35] Detalii despre funcțiile pentru numere, şiruri de caractere şi date/intervale sunt furnizate în capitolul 6.

calendaristică pot fi scrise direct: '2008-11-28' (vezi tabela INCASARI), deci, în această privință avem de unde alege.

Listing 3.6. Script PostgreSQL de creare a tabelelor BD VÎNZĂRI

```
DROP TABLE incasfact ;
DROP TABLE incasari ;
DROP TABLE liniifact ;
DROP TABLE facturi ;
DROP TABLE produse ;
DROP TABLE persclienti ;
DROP TABLE persoane ;
DROP TABLE clienti ;
DROP TABLE coduri_postale ;
DROP TABLE judete ;

CREATE TABLE judete (
    jud CHAR(2)
        CONSTRAINT pk_judete PRIMARY KEY
        CONSTRAINT ck_jud CHECK (jud=LTRIM(UPPER(jud))),
    judet VARCHAR(25)
        CONSTRAINT un_judet UNIQUE
        CONSTRAINT nn_judet NOT NULL
        CONSTRAINT ck_judet CHECK (judet=LTRIM(INITCAP(judet))),
    regiune VARCHAR(15)
        DEFAULT 'Moldova' CONSTRAINT nn_regiune NOT NULL
        CONSTRAINT ck_regiune CHECK (regiune IN ('Banat', 'Transilvania',
                'Dobrogea', 'Oltenia', 'Muntenia', 'Moldova'))
    ) ;

CREATE TABLE coduri_postale (
    codpost CHAR(6)
        CONSTRAINT pk_coduri_post PRIMARY KEY
        CONSTRAINT ck_codpost CHECK (codpost=LTRIM(codpost)),
    loc VARCHAR(25)
        CONSTRAINT nn_loc NOT NULL
        CONSTRAINT ck_loc CHECK (loc=LTRIM(INITCAP(loc))),
    jud CHAR(2) DEFAULT 'IS' CONSTRAINT fk_coduri_post_jud
        REFERENCES judete(jud) ON UPDATE CASCADE
    ) ;

CREATE TABLE clienti (
    codcl SMALLINT NOT NULL
        CONSTRAINT pk_clienti PRIMARY KEY
        CONSTRAINT ck_codcl CHECK (codcl > 1000),
    dencl VARCHAR(30)
        CONSTRAINT ck_dencl CHECK (SUBSTR(dencl,1,1) =
                UPPER(SUBSTR(dencl,1,1))),
    codfiscal CHAR(9)
        CONSTRAINT ck_codfiscal CHECK (SUBSTR(codfiscal,1,1) =
                UPPER(SUBSTR(codfiscal,1,1))),
    adresa VARCHAR(40)
        CONSTRAINT ck_adresa_clienti CHECK (SUBSTR(adresa,1,1) =
                UPPER(SUBSTR(adresa,1,1))),
    codpost CHAR(6)
        CONSTRAINT fk_clienti_cp REFERENCES coduri_postale(codpost)
                ON UPDATE CASCADE,
    telefon VARCHAR(10)
    ) ;
```

```
CREATE TABLE persoane (
    cnp CHAR(14)
        CONSTRAINT pk_persoane PRIMARY KEY
        CONSTRAINT ck_cnp CHECK (cnp=LTRIM(UPPER(cnp))),
    nume VARCHAR(20)
        CONSTRAINT ck_nume CHECK (nume=LTRIM(INITCAP(nume))),
    prenume VARCHAR(20)
        CONSTRAINT ck_prenume CHECK (prenume=LTRIM(INITCAP(prenume))),
    adresa VARCHAR(50)
        CONSTRAINT ck_adresa_persoane CHECK (SUBSTR(adresa,1,1) =
                UPPER(SUBSTR(adresa,1,1))),
    sex CHAR(1) DEFAULT 'B'
        CONSTRAINT ck_sex CHECK (sex IN ('F','B')),
    codpost CHAR(6)
        CONSTRAINT fk_persoane_cp REFERENCES coduri_postale(codpost)
                ON UPDATE CASCADE,
    telacasa VARCHAR(10),
    telbirou VARCHAR(10),
    telmobil VARCHAR(10),
    email VARCHAR(30)
    ) ;

CREATE TABLE persclienti (
    cnp CHAR(14)
        CONSTRAINT fk_persclienti_persoane REFERENCES persoane(cnp)
                ON UPDATE CASCADE,
    codcl SMALLINT
        CONSTRAINT fk_persclienti_clienti REFERENCES clienti(codcl)
                ON UPDATE CASCADE,
    functie VARCHAR(25)
        CONSTRAINT ck_functie CHECK (SUBSTR(functie,1,1) =
                UPPER(SUBSTR(functie,1,1))),
        CONSTRAINT pk_persclienti PRIMARY KEY (cnp, codcl, functie)
    ) ;

CREATE TABLE produse (
    codpr SMALLINT NOT NULL
        CONSTRAINT pk_produse PRIMARY KEY
        CONSTRAINT ck_codpr CHECK (codpr > 0),
    denpr VARCHAR(30) CONSTRAINT ck_denpr
        CHECK (SUBSTR(denpr,1,1) = UPPER(SUBSTR(denpr,1,1))),
    um VARCHAR(10),
    grupa VARCHAR(15) CONSTRAINT ck_produse_grupa
        CHECK (SUBSTR(grupa,1,1) = UPPER(SUBSTR(grupa,1,1))),
    procTVA NUMERIC(2,2) DEFAULT .19
    ) ;

CREATE TABLE facturi (
    nrfact INTEGER
        CONSTRAINT pk_facturi PRIMARY KEY,
    datafact DATE DEFAULT CURRENT_DATE,
        CONSTRAINT ck_datafact CHECK (datafact >= TO_DATE('2007-01-01',
                'YYYY-MM-DD')     AND datafact <= DATE'2015-12-31'),
    codcl SMALLINT
        CONSTRAINT fk_facturi_clienti REFERENCES clienti(codcl)
                ON UPDATE CASCADE,
    obs VARCHAR(50)
    ) ;
```

```
CREATE TABLE liniifact (
    nrfact INTEGER
        CONSTRAINT fk_liniifact_facturi REFERENCES facturi(nrfact)
                ON UPDATE CASCADE,
    linie SMALLINT CONSTRAINT ck_linie CHECK (linie > 0),
    codpr SMALLINT
        CONSTRAINT fk_liniifact_produse REFERENCES produse(codpr) ON UPDATE
CASCADE,
    cantitate NUMERIC(8),
    pretunit NUMERIC (9,2),
    CONSTRAINT pk_liniifact PRIMARY KEY (nrfact,linie)
    ) ;

CREATE TABLE incasari (
    codinc BIGINT NOT NULL
        CONSTRAINT pk_incasari PRIMARY KEY,
    datainc DATE DEFAULT CURRENT_DATE
        CONSTRAINT ck_datainc CHECK (datainc >= '2007-01-01'
                AND datainc <= TO_DATE('31/12/2015','DD/MM/YYYY')),
    coddoc CHAR(4)
        CONSTRAINT ck_coddoc CHECK(coddoc=UPPER(LTRIM(coddoc))),
    nrdoc VARCHAR(16),
    datadoc DATE DEFAULT CURRENT_DATE - 7
        CONSTRAINT ck_datadoc CHECK (datadoc >= DATE'2007-01-01'
                AND datadoc <= DATE'2010-12-31')
    ) ;

CREATE TABLE incasfact (
    codinc BIGINT CONSTRAINT fk_incasfact_incasari
        REFERENCES incasari(codinc) ON UPDATE CASCADE,
    nrfact INTEGER CONSTRAINT fk_incasfact_facturi
        REFERENCES facturi(nrfact) ON UPDATE CASCADE,
    transa NUMERIC(11,2) NOT NULL,
        CONSTRAINT pk_incasfact PRIMARY KEY (codinc, nrfact)
    ) ;

ALTER TABLE incasari ADD CONSTRAINT ck_date_incdoc CHECK (DataInc >= DataDoc) ;
```

3.3.12. Scripturi de creare a tabelelor în SQL Server

SQL Server a fost (şi încă mai este în oarecare măsură) cel mai puțin fidel standardelor SQL dintre cele patru servere BD pe care le folosim în această carte. Astfel, numai în privința datelor de tip întreg, există patru tipuri:

- BIGINT, cu poate valori cuprinse -2^{63} între şi 2^{63} -1;
- INT, cu poate valori cuprinse -231 între şi 231 -1;
- SMALLINT, cu poate valori cuprinse -215 între şi 215 -1;
- TINYINT, cu poate valori cuprinse 0 şi 255;

În cele patru tabele ale BD TRIAJ am folosit tipul SMALLINT pentru atributul IdDoctor, INTEGER pentru atributul IdPacient şi BIGINT pentru IdExaminare.

Pentru date calendaristice, SQL Server are două tipuri, SMALLDATETIME şi DATETIME. Ambele sunt, de fapt, apropiate de tipul TIMESTAMP din standard.

Diferența ține de precizie. În timp ce o valoare SMALLDATETIME are o "precizie" de ordinul minutelor (altfel spus, conține: an, lună, zi, oră, minut), valorile DATETIME au o acuratețe de 3,33 milisecunde (conțin, plus, secunde şi fracțiuni de milisecunde). Tipul TIMESTAMP are altă funcționalitate decât în standard şi celelalte servere BD. Versiunea SQL Server 2008 introduce tipul DATETIME, dar noi vom rula aproape toate exemplele în SQL Server 2005.

Listingul 3.7 este echivalentul SQL Server al listingurilor 3.1, 3.3 şi 3.5. Ca tip de dată pentru atributele Început_Gardă şi Sfârşit_Gardă (tabela GĂRZI) am ales tipul SMALLDATETIME, întrucât precizia de ordinul minutelor este suficientă, iar pentru atributul DataOra_Examinare am optat pentru DATETIME.

Atributele pentru care dorim ca valorile să fie unice şi generate automat trebuie declarate în comanda CREATE TABLE cu opțiunea IDENTITY. În lipsa oricărui paramentru (cazul nostru), prima valoare generată este 1 iar incrementul este tot 1, valoarea maximă fiind limitată doar de tipul ales (TINYINT, SMALLINT, INTEGER, BIGINT).

Listing 3.7. Script SQL Server de creare a tabelelor BD TRIAJ

```
DROP TABLE triaj ;
DROP TABLE pacienti ;
DROP TABLE garzi ;
DROP TABLE doctori ;

CREATE TABLE doctori (
    iddoctor SMALLINT IDENTITY
        CONSTRAINT pk_doctori PRIMARY KEY,
    numedoctor VARCHAR(50) NOT NULL,
    specialitate VARCHAR(40) NOT NULL,
    datanasterii SMALLDATETIME
    );

CREATE TABLE garzi (
    iddoctor SMALLINT NOT NULL
        REFERENCES doctori (iddoctor) ON UPDATE CASCADE,
    inceput_garda SMALLDATETIME
        DEFAULT CURRENT_TIMESTAMP
        NOT NULL,
    sfirsit_garda SMALLDATETIME,
    CONSTRAINT pk_garzi PRIMARY KEY (iddoctor, inceput_garda)
    ) ;

CREATE TABLE pacienti (
    idpacient INTEGER IDENTITY
        CONSTRAINT pk_pacienti PRIMARY KEY,
    numepacient VARCHAR(60) NOT NULL,
    CNP NUMERIC(13),
    adresa VARCHAR(100),
    loc VARCHAR(30),
    judet CHAR(2),
    tara VARCHAR(30) DEFAULT 'Romania',
    serie_nr_act_identitate VARCHAR(20)
    );

CREATE TABLE triaj (
    idexaminare BIGINT IDENTITY CONSTRAINT pk_internari PRIMARY KEY,
```

```
        dataora_examinare DATETIME DEFAULT CURRENT_TIMESTAMP NOT NULL,
        idpacient INTEGER NOT NULL
            CONSTRAINT fk_internari_pacienti
                    REFERENCES pacienti (idpacient) ON UPDATE CASCADE,
        simptome VARCHAR(500) NOT NULL,
        tratament_imediat VARCHAR(500),
        sectie_destinatie VARCHAR(30)
        );
```

Lansarea în execuție (folosind modulul standard *SQL Server Management Studio*) a comenzilor din acest listing este ilustrată în figura 3.5.

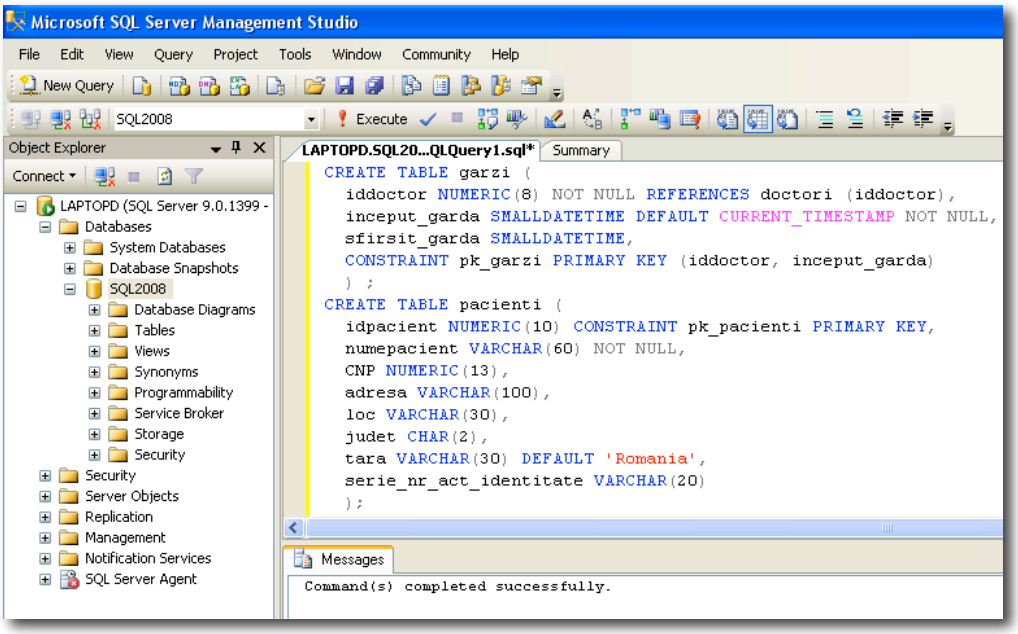


Figura 3.5. Lansarea în execuţie în SQL Server a scriptului (comenzilor) de creare a tabelelor BD TRIAJ

În privința comenzilor de (ştergere şi) creare a tabelelor din BD VÎNZĂRI (listing 3.8), am mai adăuga că:

- similar DB2, nu există o funcție cu funcționalitatea INITCAP (din Oracle şi PostgreSQL)
- constantele de tip dată calendaristică se scriu ca în DB2, fără "prefixul" DATE;
- pentru atributul Linie din LINIIFACT am folosit tipul TINYINT;
- funcția care preia data (şi ora) sistemului este GETDATE();
- valoarea de început a atributului CodCl din tabela CLIENȚI este 1001, aşa că la creare am folosit clauza IDENTITY (1001,1); analog am procedat cu atributul CodInc din tabela ÎNCASARI, a cărui valoare de start este 1234.

Listing 3.8. Script SQL Server de creare a tabelelor BD VÎNZĂRI

```
DROP TABLE incasfact  ;
DROP TABLE incasari ;
DROP TABLE liniifact ;
DROP TABLE facturi ;
DROP TABLE produse ;
DROP TABLE persclienti ;
DROP TABLE persoane ;
DROP TABLE clienti ;
DROP TABLE coduri_postale ;
DROP TABLE judete ;

CREATE TABLE judete (
    Jud CHAR(2)
        CONSTRAINT pk_judete PRIMARY KEY
        CONSTRAINT ck_jud CHECK (Jud=LTRIM(UPPER(Jud))),
    Judet VARCHAR(25)
        CONSTRAINT un_judet UNIQUE
        CONSTRAINT nn_judet NOT NULL,
    Regiune VARCHAR(15)
        DEFAULT 'Moldova' CONSTRAINT nn_regiune NOT NULL
        CONSTRAINT ck_regiune CHECK (Regiune IN ('Banat', 'Transilvania',
                'Dobrogea', 'Oltenia', 'Muntenia', 'Moldova'))
    ) ;

CREATE TABLE coduri_postale (
    CodPost CHAR(6)
        CONSTRAINT pk_coduri_post PRIMARY KEY
        CONSTRAINT ck_codpost CHECK (CodPost=LTRIM(CodPost)),
    Loc VARCHAR(25) CONSTRAINT nn_loc NOT NULL,
    Jud CHAR(2) DEFAULT 'IS'
        CONSTRAINT fk_coduri_post_jud REFERENCES judete(Jud) ON UPDATE CASCADE
    ) ;

CREATE TABLE clienti (
    CodCl SMALLINT IDENTITY (1001,1)
        CONSTRAINT pk_clienti PRIMARY KEY
        CONSTRAINT ck_codcl CHECK (CodCl > 1000),
    DenCl VARCHAR(30) NOT NULL
        CONSTRAINT ck_dencl CHECK (SUBSTRING(DenCl,1,1) =
                UPPER(SUBSTRING(DenCl,1,1))),
    CodFiscal CHAR(9) NOT NULL
        CONSTRAINT un_codfiscal UNIQUE
        CONSTRAINT ck_codfiscal CHECK (SUBSTRING(CodFiscal,1,1) =
                UPPER(SUBSTRING(CodFiscal,1,1))),
    Adresa VARCHAR(40)
        CONSTRAINT ck_adresa_clienti CHECK (SUBSTRING(Adresa,1,1) =
                UPPER(SUBSTRING(Adresa,1,1))),
    CodPost CHAR(6) NOT NULL
        CONSTRAINT fk_clienti_cp
                REFERENCES coduri_postale(CodPost) ON UPDATE CASCADE,
    Telefon VARCHAR(10)
    ) ;

CREATE TABLE persoane (
    CNP CHAR(14)
        CONSTRAINT pk_persoane PRIMARY KEY
        CONSTRAINT ck_cnp CHECK (CNP=LTRIM(UPPER(CNP))),
    Nume VARCHAR(20) NOT NULL,
```

```
    Prenume VARCHAR(20) NOT NULL,
    Adresa VARCHAR(50)
        CONSTRAINT ck_adresa_persoane CHECK (SUBSTRING(Adresa,1,1) =
                UPPER(SUBSTRING(Adresa,1,1))),
    Sex CHAR(1) DEFAULT 'B' NOT NULL
        CONSTRAINT ck_sex CHECK (Sex IN ('F','B')),
    CodPost CHAR(6) NOT NULL CONSTRAINT fk_persoane_cp
        REFERENCES coduri_postale(CodPost) ON UPDATE CASCADE,
    TelAcasa VARCHAR(10),
    TelBirou VARCHAR(10),
    TelMobil VARCHAR(10),
    Email VARCHAR(30)
    ) ;

CREATE TABLE persclienti (
    CNP CHAR(14) NOT NULL
        CONSTRAINT fk_persclienti_persoane REFERENCES persoane(CNP)
                ON UPDATE CASCADE,
    CodCl SMALLINT NOT NULL
        CONSTRAINT fk_persclienti_clienti REFERENCES clienti(CodCl)
    -- aici ON UPDATE CASCADE nu functioneaza, adica genereaza eroare !!!!!!! --
    ,
    Functie VARCHAR(25) NOT NULL
        CONSTRAINT ck_functie CHECK (SUBSTRING(Functie,1,1) =
                UPPER(SUBSTRING(Functie,1,1))),
    CONSTRAINT pk_persclienti PRIMARY KEY (CNP, CodCl, Functie)
    ) ;

CREATE TABLE produse (
    CodPr SMALLINT IDENTITY
        CONSTRAINT pk_produse PRIMARY KEY
        CONSTRAINT ck_codpr CHECK (CodPr > 0),
    DenPr VARCHAR(30) NOT NULL CONSTRAINT ck_denpr
        CHECK (SUBSTRING(DenPr,1,1) = UPPER(SUBSTRING(DenPr,1,1))),
    UM VARCHAR(10) NOT NULL,
    Grupa VARCHAR(15)
        CONSTRAINT ck_produse_grupa
                CHECK (SUBSTRING(Grupa,1,1) = UPPER(SUBSTRING(Grupa,1,1))),
    ProcTVA NUMERIC(2,2) DEFAULT .19 NOT NULL
    ) ;

CREATE TABLE facturi (
    NrFact INTEGER
        CONSTRAINT pk_facturi PRIMARY KEY,
    DataFact SMALLDATETIME DEFAULT GETDATE() NOT NULL,
        CONSTRAINT ck_datafact CHECK (DataFact >= '2007-01-01'
                AND DataFact <= '2015-12-31'),
    CodCl SMALLINT NOT NULL
        CONSTRAINT fk_facturi_clienti
                REFERENCES clienti(CodCl) ON UPDATE CASCADE,
    Obs VARCHAR(50)
    ) ;

CREATE TABLE liniifact (
     NrFact INTEGER
        CONSTRAINT fk_liniifact_facturi
                REFERENCES facturi(NrFact) ON UPDATE CASCADE,
    Linie TINYINT NOT NULL
        CONSTRAINT ck_linie CHECK (Linie > 0),
```

```
    CodPr SMALLINT NOT NULL
        CONSTRAINT fk_liniifact_produse
            REFERENCES produse(CodPr) ON UPDATE CASCADE,
    Cantitate NUMERIC(8) NOT NULL,
    PretUnit NUMERIC (9,2) NOT NULL,
    CONSTRAINT pk_liniifact PRIMARY KEY (NrFact,Linie)
    ) ;

CREATE TABLE incasari (
    CodInc BIGINT IDENTITY (1234,1)
        CONSTRAINT pk_incasari PRIMARY KEY,
    DataInc SMALLDATETIME DEFAULT GETDATE() NOT NULL
        CONSTRAINT ck_datainc CHECK (DataInc >= '2007-01-01' AND
            DataInc <= '2015-12-31'),
    CodDoc CHAR(4) NOT NULL
        CONSTRAINT ck_coddoc CHECK(CodDoc=UPPER(LTRIM(CodDoc))),
    NrDoc VARCHAR(16) NOT NULL,
    DataDoc SMALLDATETIME DEFAULT GETDATE() - 7 NOT NULL
        CONSTRAINT ck_datadoc CHECK (DataDoc >= '2007-01-01'
            AND DataDoc <= '2015-12-31')
    ) ;

CREATE TABLE incasfact (
    CodInc BIGINT
        CONSTRAINT fk_incasfact_incasari
            REFERENCES incasari(CodInc) ON UPDATE CASCADE,
    NrFact INTEGER
        CONSTRAINT fk_incasfact_facturi
            REFERENCES facturi(NrFact) ON UPDATE CASCADE,
    Transa NUMERIC(11,2) NOT NULL,
    CONSTRAINT pk_incasfact PRIMARY KEY (CodInc, NrFact)
    ) ;

ALTER TABLE incasari ADD CONSTRAINT ck_date_incdoc CHECK (DataInc >= DataDoc) ;
```

Una dintre cele mai neplăcute idei ale autorilor SQL Server este case sensitive-ul dus la extrem. Dacă numele unui atribut a fost scris, la CREATE TABLE cu litere mici, toate comenzile ALTER TABLE, INSERT, UPDATE, DELETE, SELECT următoare trebuie să facă referință la atributul respectiv exclusiv cu litere mici !!! Aşa că atenție la modul de scriere a numelui atributelor din acest script! Dacă în ultima comandă – ALTER TABLE, în loc de *DataInc* am fi scris *datainc*, ne-am fi procopsit cu un mesaj de eroare.

3.4. Modificarea conținutului

În acest moment, am avem definită, în mare, structura bazei de date. Pasul următor, care se derulează pe toată durata vieții (folosirii efective) unei BD este actualizarea conținului. În BD TRIAJ trebuie să introducem efectiv datele despre medicii clinicii, pacienții şi evenimentele de la camera de primire (triaj) a bolnavilor.

SQL prezintă comenzi dedicate modificării conținutului unei tabele, înțelegând prin aceasta trei acțiuni prin care se actualizează baza:

a) adăugarea de noi linii la cele existente în tabelă,
b) ştergerea unor linii,
c) modificarea valorii unui atribut.

3.4.1. Adăugarea de linii în tabele

Comanda SQL de adăugare de noi linii este INSERT, cu un format destul de simplu. Iată câteva exemple:

```
INSERT INTO judete VALUES ('IS', 'Iasi', 'Moldova') ;
INSERT INTO coduri_postale VALUES ('700505', 'Iasi', 'IS') ;
INSERT INTO clienti VALUES (1001, 'Client 1 SRL', 'R1001', 'Tranzitiei, 13 bis',
        '700505', NULL) ;
INSERT INTO persoane VALUES ('CNP1', 'Ioan', 'Vasile',
        'I.L.Caragiale, 22', 'B', '700505', '123456', '987654', '094222222', NULL) ;
INSERT INTO persclienti VALUES ('CNP1', 1001, 'Director general');
INSERT INTO produse VALUES (1, 'Produs 1','buc', 'Tigari', .19) ;
INSERT INTO facturi (NrFact, DataFact, CodCl)  VALUES (1111, DATE'2008-01-01', 1001);
INSERT INTO facturi (NrFact, DataFact, CodCl)
        VALUES (1121, DATE'2008-08-07', 1001);
INSERT INTO liniifact (nrfact, linie, codpr, cantitate, pretunit)
        VALUES (1111, 1, 1, 50, 1000) ;
INSERT INTO liniifact (nrfact, linie, codpr, cantitate, pretunit)
        VALUES (1111, 2, 2, 75, 1050) ;

INSERT INTO incasari VALUES (1234, DATE'2008-08-15', 'OP', '111', DATE'2008-10-08') ;
INSERT INTO incasfact VALUES (1234, 1111, 53996) ;
```

Ordinea valorilor din clauza VALUES trebuie să fie identică cu cea declarată la crearea (sau modificarea structurii) tabelelor. Modificarea este posibilă numai prin enumerarea, după numele tabelei, a atributelor ce vor primi valorile specificate. Pentru tabelele FACTURI şi LINIIFACT au fost incluse în clauza VALUES mai puține valori decât atributele tabelei. Este obligatorie, în aceste cazuri, precizarea atributelor care vor primi valorile. Restul, adică cele nespecificate, vor avea, pe liniile respective, valori NULL.

Dacă valorile specificate în clauza VALUES vin în contradicție cu măcar o restricție declarată, se afişează un mesaj de eroare, iar comanda INSERT este respinsă. Spre exemplu, *INSERT INTO judete VALUES ('IS', 'Iasi', 'Mldova')* încalcă restricția definită pentru atributul Regiune (CK_REGIUNE), iar rezultatul din DB2 este prezentat în figura 3.6.

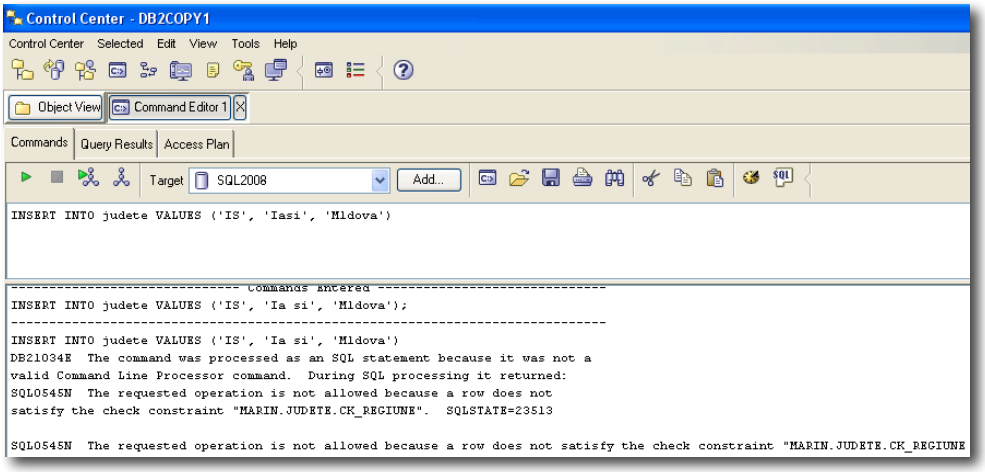


Figura 3.6. Violarea (în DB2) restricţiei CK_REGIUNE

3.4.2. Ştergerea liniilor

Comanda SQL pentru ştergerea uneia sau mai multor linii dintr-o tabelă este DELETE, al cărei format general este: *DELETE [FROM] nume-tabelă [WHERE predicat]*. Din tabelă vor fi şterse toate liniile care îndeplinesc condiţia specificată în predicatul din clauza WHERE. Astfel, *DELETE FROM facturi WHERE NrFact = 1122* elimină din tabela FACTURI toate înregistrările în care valoarea atributului NrFact este egală cu 1122, altfel spus, din tabela FACTURI este ştearsă linia care se referă la factura 1122. Când pentru această factură există înregistrări copil în LINIIFACT sau INCASFACT, SGBD-ul ţine cont de opţiunea declarată la crearea tabelei. Dacă s-a specificat (implicit sau explicit) ON DELETE RESTRICT, ştergerea este interzisă. În cazul ON DELETE CASCADE sunt şterse, odată cu linia din FACTURI, toate liniile copil din LINIIFACT şi INCASFACT.

Comanda *DELETE FROM judete WHERE Regiune = 'Moldova'* elimină din baza de date toate judeţele din Moldova (şterge toate liniile tabelei JUDETE în care valoarea atributului Regiune este Moldova).

3.4.3. Modificarea valorilor unor atribute

Pentru a modifica valoarea unuia sau mai multor atribute pe una sau mai multe linii dintr-o tabelă se foloseşte comanda UPDATE cu formatul general (simplificat): *UPDATE tabelă SET atribut1 = expresie1 [, atribut2= expresie2 ….] WHERE predicat*. Modificarea se va produce pe toate liniile tabelei care îndeplinesc condiţia formulată prin predicat. Exemple:

- Noul număr de telefon al clientului ce are codul 1001 este 0232-313131:

UPDATE CLIENTI SET Telefon = '0232-313131' WHERE CodCl = 1001

- În cadrul unei noi relaxări fiscale, se decide creşterea procentului TVA de la 19% la 22% pentru toate produsele. Atributul modificat este ProcTVA din tabela PRODUSE, pe toate liniile:

UPDATE PRODUSE SET ProcTVA = .22

3.4.4. Scripturile de populare în DB2

În practică, popularea cu înregistrări se face interactiv, folosind meniuri şi formulare prin care utilizatorii (de obicei, total nepricepuți în ale bazelor de date) preiau informațiile. Noi însă, neştiind să construim formulare, vom adăuga/modifica/şterge "manual" înregistrări în/din tabele, folosind cele trei comenzi pe care le-am mai pomenit, INSERT, UPDATE şi DELETE.

Spuneam, la crearea tabelelor, că atributele cu valori auto-generate au un regim special la editare. Astfel, în tabela DOCTORI, IdDoctor este un asemenea atribut. Dacă am urma logica INSERT-ului, am fi tentați să scriem:

INSERT INTO doctori VALUES (1, 'Vasilcu Ionel', 'chirurgie', '1965-11-11') ;

Interesul nostru ține de faptul că, la crearea tabelei, am declarat IdDoctor ca având valori generate, în timp ce acum, vrem să atribuim acestui atribut valoarea 1 prin INSERT. Chiar dacă valoarea declarată în comanda INSERT coincide cu valoarea generată automat (adică 1), serverul se va împotrivi – vezi figura 3.7.

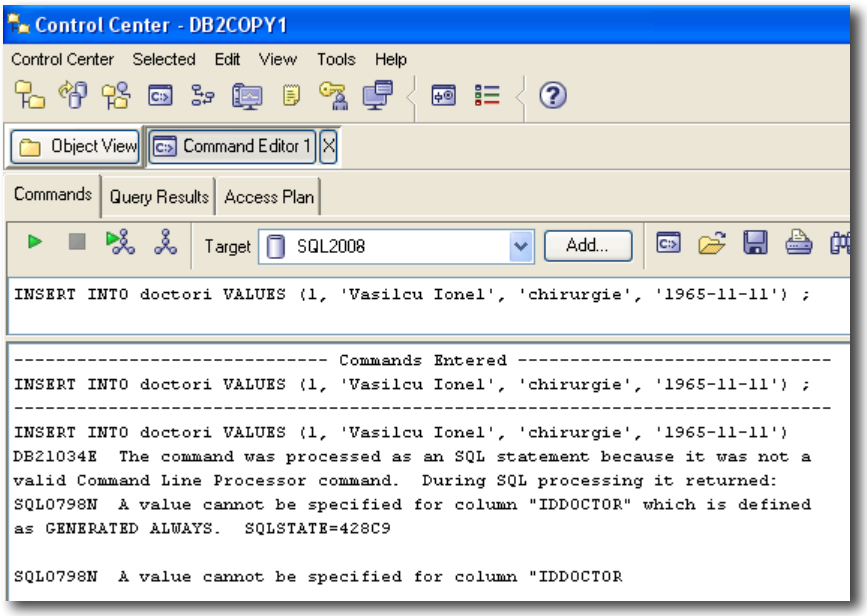


Figura 3.7. Eroare (DB2) din cauza încercării de stabilire prin INSERT a valorii unui atribut cu valori auto-generate.

Reacția serverului este îndreptățită. Listingul 3.9 conține scriptul de populare a celor patru tabele din BD TRIAJ. Comenzile DELETE de început asigură "curățarea" tabelelor, în caz că aveau vreo înregistrare dinainte. În lipsa acestora, la

următoarea lansare a scriptului, s-ar fi violat restricţia de cheie primară pentru tabela GĂRZI. Despre a comanda de final, COMMIT, vom discuta peste câteva paragrafe.

Listing 3.9. Script DB2 de populare a tabelelor BD TRIAJ

```
DELETE FROM triaj ;
DELETE FROM pacienti ;
DELETE FROM  garzi ;
DELETE FROM  doctori ;

INSERT INTO doctori (numedoctor, specialitate, datanasterii)
    VALUES ('Vasilcu Ionel', 'chirurgie', '1965-11-11') ;
INSERT INTO doctori (numedoctor, specialitate, datanasterii)
    VALUES ('Georgescu Mircea', 'hepatologie', '1966-01-12') ;
INSERT INTO doctori (numedoctor, specialitate, datanasterii)
    VALUES ('Zahir Tudorel', 'chirurgie', '1965-11-11') ;
INSERT INTO doctori (numedoctor, specialitate, datanasterii) VALUES
    ('Bostan Vasile', 'cardiologie', '1965-11-11') ;
INSERT INTO doctori (numedoctor, specialitate, datanasterii)
    VALUES ('Popescu Ionela', 'boli interne', '1965-11-11') ;

INSERT INTO garzi VALUES (2, '2008-01-03 7:00:00', '2008-01-03 19:00:00') ;
INSERT INTO garzi VALUES (3, '2008-01-03 19:00:00', '2008-01-04 7:00:00') ;
INSERT INTO garzi VALUES (4, '2008-01-04 07:00:00', '2008-01-04 18:00:00') ;
INSERT INTO garzi VALUES (5, '2008-01-04 18:00:00', '2008-01-05 06:00:00') ;
INSERT INTO garzi VALUES (2, '2008-01-05 6:00:00', '2008-01-05 14:00:00') ;
INSERT INTO garzi VALUES (3, '2008-01-05 14:00:00', '2008-01-05 22:00:00') ;
INSERT INTO garzi VALUES (5, '2008-01-05 22:00:00', '2008-01-06 9:00:00') ;

INSERT INTO pacienti (numepacient, cnp, adresa, serie_nr_act_identitate) VALUES
    ('Stroe Mihaela', 2601228390834, 'Bd. Cantemir, 32, Bl.G4, Sc.C, Ap.4', 'MR366766');
INSERT INTO pacienti (numepacient, cnp, adresa, serie_nr_act_identitate) VALUES
    ('Buzatu Corneliu', 1650512370514, 'Str. Desprimaveririi, 112', 'MX456783') ;
INSERT INTO pacienti (numepacient, cnp, adresa, serie_nr_act_identitate) VALUES
     ('Spineanu Marius', 5010101380625, 'Bd. Stefan cel Mare, 4, Bl.I1, Sc.A, Ap.24', 'MX213345') ;
INSERT INTO pacienti (numepacient, cnp, adresa, serie_nr_act_identitate) VALUES
    ('Bagdasar Adela', 2601002250611, 'Str. Primaverii, 17', 'MX345678') ;
INSERT INTO pacienti (numepacient, cnp, adresa, serie_nr_act_identitate) VALUES
    ('Ionascu Ionelia', 6001122390199, 'Str. Florilor', 'MX654322') ;
INSERT INTO pacienti (numepacient, cnp, adresa, serie_nr_act_identitate) VALUES
    ('Popescu Maria-Mirabela', 2721231300888,
        'Bd. 22 Decembrie, 2, Bl.5, Sc.B, Ap.21', 'MX765432');
INSERT INTO pacienti (numepacient, cnp, adresa, serie_nr_act_identitate) VALUES
    ('Stroescu Mihaela Oana', 6020719120545, 'Str. Lenei Nr. 234', 'MX876567') ;
INSERT INTO pacienti (numepacient, cnp, adresa, serie_nr_act_identitate) VALUES
    ('Cazan Ana Maria', 2690202200345, 'Bd. Independentei, 87, Bl.K3, Sc.A, Ap.34', 'MX987789');

INSERT INTO triaj (dataora_examinare, idpacient, simptome, tratament_imediat, sectie_destinatie)
    VALUES ('2008-01-03 7:18:00', 1, 'dureri de stomac intense', NULL, 'boli interne');
INSERT INTO triaj (dataora_examinare, idpacient, simptome, tratament_imediat, sectie_destinatie)
    VALUES ('2008-01-03 8:45:00', 2, 'febra puternica, varsaturi', 'penicilina', 'boli interne');
INSERT INTO triaj (dataora_examinare, idpacient, simptome, tratament_imediat, sectie_destinatie)
    VALUES ('2008-01-03 12:45:00', 3, 'deranjament stomacal', 'scobutil', NULL);
INSERT INTO triaj (dataora_examinare, idpacient, simptome, tratament_imediat, sectie_destinatie)
    VALUES ('2008-01-03 20:45:00', 4, 'palpitatii cardiace', 'linistin', 'cardiologie');
INSERT INTO triaj (dataora_examinare, idpacient, simptome, tratament_imediat, sectie_destinatie)
    VALUES ('2008-01-04 1:28:00', 5, 'plaga profunda picior drept', 'antibiotice, pansament', NULL);
INSERT INTO triaj (dataora_examinare, idpacient, simptome, tratament_imediat, sectie_destinatie)
```

```
    VALUES ('2008-01-04 10:45:00', 3, 'contractii stomacale, varsaturi', NULL, 'boli interne');
INSERT INTO triaj (dataora_examinare, idpacient, simptome, tratament_imediat, sectie_destinatie)
    VALUES ('2008-01-04 11:20:00', 7, 'fata de culoare galbena, ameteli', NULL, 'hepatologie');
INSERT INTO triaj (dataora_examinare, idpacient, simptome, tratament_imediat, sectie_destinatie)
    VALUES ('2008-01-04 22:45:00', 8, 'dureri articulare', 'scobutil', NULL);
INSERT INTO triaj (dataora_examinare, idpacient, simptome, tratament_imediat, sectie_destinatie)
    VALUES ('2008-01-05 6:18:00', 5, 'febra puternica, delir', 'penicilina', 'chirurgie');
COMMIT;
```

Ei bine, chiar dacă scriptul începe cu ştergerea eventualelor linii din cele patru tabele, la a doua (a treia,...) lansare în execuţie tot nu scăpăm de probleme – vezi figura 3.8.

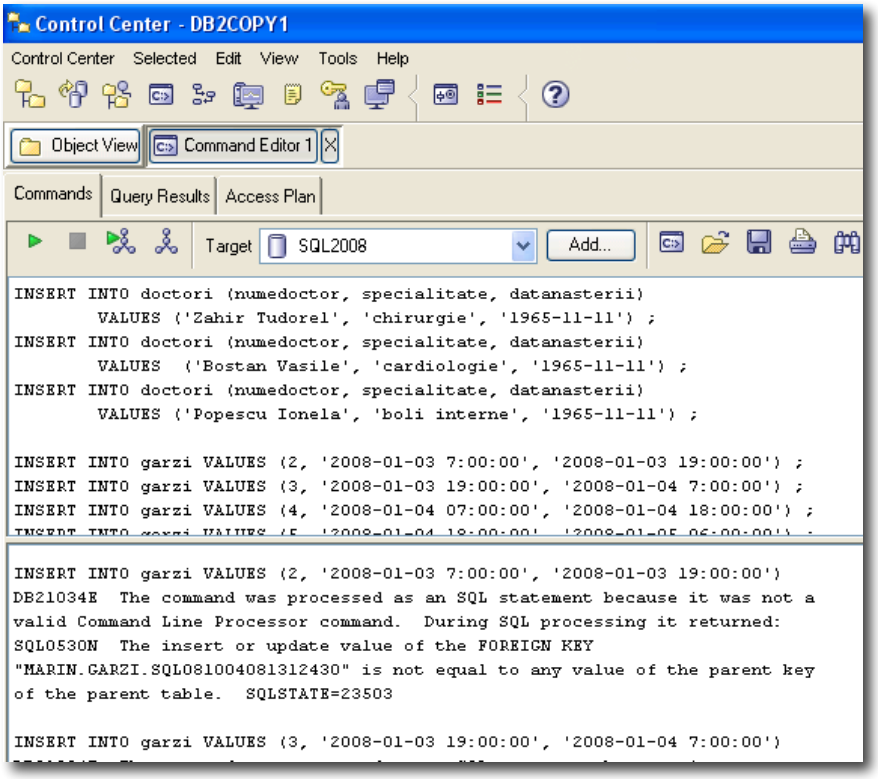


Figura 3.8. Necazuri (DB2) cu atributele ale căror valori sunt generate automat

Această eroare dezvăluie cea mai importantă durere de cap cu atributele auto-generate, şi anume că nu le putem controla. Astfel, la a doua lansare, chiar dacă toate liniile au fost şterse, valorile de la 1 la 5 pentru IdDoctor au fost consumate, aşa că, acum, id-urile celor cinci medici sunt de la 6 la 10. Baiul derivă de la faptul că IdDoctor este cheie străină în tabela GĂRZI. Ori, şi la a doua lansare a scriptului, prima comandă de inserare în GĂRZI rămâne tot *INSERT INTO garzi VALUES (2, '2008-01-03 7:00:00', '2008-01-03 19:00:00')*, iar valoarea 2 nu se mai regăseşte în tabela părinte (DOCTORI).

Cum rezolvăm această problemă enervantă ? Soluția cea mai elegantă este ca, la inserarea în tabelele copil, valorile cheilor străine care reprezintă (în tabela părinte) atribute cu valori generate automat, să nu fie specificate prin constante, ci prin subsconsultări:

```
INSERT INTO garzi VALUES ( (SELECT IdDoctor FROM doctori WHERE
        numedoctor='Vasilcu Ionel'), '2008-01-03 7:00:00', '2008-01-03 19:00:00') ;
```

Din păcate, nivelul nostru de SQL este, în acest paragraf, mult sub cerințele acestei sintaxe, aşa încât nu ne rămâne decât soluția ruşinoasă, adică re-lansarea scriptului de creare (listing 3.1), şi apoi a celui de populare (listing 3.9) doar o singură dată. Ar mai fi, însă şi o a treia soluție, pe care o folosim în scriptul de populare a tabelelor bazei de date VÎNZĂRI – listing 3.10. DB2 pune la dispoziție funcția IDENTITY_VAL_LOCAL care "ține minte" ultima valoare a unei coloane auto-generate. În aceste condiții, după fiecare înregistrare părinte în care cheia primară este auto-generată, vom insera toate înregistrările copil ale acesteia, ceea ce este destul de neplăcut. Iar dacă am fi avut în bază o tabelă cu două chei străine, chei străine auto-generate în tabelele părinte, chiar că lucrurile luau o întorsătură interesantă.

O premieră în listingul 3.10 ține de posibilitatea oferită de DB2 ca într-o clauză VALUES a comenzii INSERT să fie specificate, simultan, mai multe linii. Astfel, cele şase județe sunt introduse în tabelă cu o singură comandă. De fapt, majoritatea liniilor introduse în tabele folosesc această facilitate.

Listing 3.10. Script DB2 de populare a tabelelor BD VÎNZĂRI

```
DELETE FROM incasfact ;
DELETE FROM incasari ;
DELETE FROM liniifact ;
DELETE FROM facturi ;
DELETE FROM produse ;
DELETE FROM persclienti ;
DELETE FROM persoane ;
DELETE FROM clienti ;
DELETE FROM coduri_postale ;
DELETE FROM judete ;

INSERT INTO judete VALUES  ('IS', 'Iasi', 'Moldova'), ('VN', 'Vrancea', 'Moldova'),
    ('NT', 'Neamt', 'Moldova'), ('SV', 'Suceava', 'Moldova'),
    ('VS', 'Vaslui', 'Moldova'), ('TM', 'Timis', 'Banat')  ;

INSERT INTO coduri_postale VALUES ('700505', 'Iasi', 'IS'), ('700510', 'Iasi', 'IS'),
    ('700515', 'Iasi', 'IS'), ('701150', 'Pascani', 'IS'), ('706500', 'Vaslui', 'VS'),
    ('706510', 'Vaslui', 'VS'), ('705300', 'Focsani', 'VN'), ('705310', 'Focsani', 'VN'),
    ('706400', 'Birlad', 'VS'), ('705800', 'Suceava', 'SV'), ('705550', 'Roman', 'NT'),
    ('701900', 'Timisoara', 'TM') ;

INSERT INTO persoane VALUES
    ('CNP1', 'Ioan', 'Vasile', 'I.L.Caragiale, 22', 'B', '700505', 123456', '987654', '094222222', NULL),
    ('CNP2', 'Vasile', 'Ion', NULL,          'B', '700505', '234567', '876543', '094222223', 'Ion@a.ro'),
    ('CNP3', 'Popovici', 'Ioana', 'V.Micle, Bl.I, Sc.B,Ap.2', 'F', '701150', '345678', NULL, '094222224',
        NULL) ;
INSERT INTO persoane VALUES
    ('CNP4', 'Lazar', 'Caraion', 'M.Eminescu, 42', 'B', '706500', '456789', NULL, '094222225', NULL),
```

```
      ('CNP5', 'Iurea', 'Simion', 'I.Creanga, 44 bis', 'B', '706500', '567890', '543210', NULL, NULL),
      ('CNP6', 'Vasc', 'Simona', 'M.Eminescu, 13', 'F', '701150', NULL, '432109', '094222227', NULL),
      ('CNP7', 'Popa', 'Ioanid', 'I.Ion, Bl.H2, Sc.C, Ap.45', 'B', '701900', '789012', '321098', NULL, NULL),
      ('CNP8', 'Bogacs', 'Ildiko', 'I.V.Viteazu, 67', 'F', '705550', '890123', '210987', '094222229', NULL),
           ('CNP9', 'Ioan', 'Vasilica', 'Garii, Bl.B4, Sc.A, Ap.1', 'F','701900', '901234', '109876', '094222230',
                NULL);

-- primul client:
INSERT INTO clienti VALUES (DEFAULT, 'Client 1 SRL', 'R1001', 'Tranzitiei, 13 bis',
      '700505', NULL) ;
-- liniile copil din tabelele PERSCLIENTI si FACTURI:
INSERT INTO persclienti VALUES ('CNP1', identity_val_local(), 'Director general');
INSERT INTO facturi (NrFact, DataFact, CodCl) VALUES
      (1111, '2007-08-01', identity_val_local()), (1115,'2007-08-02', identity_val_local()),
      (1117, '2007-08-03', identity_val_local()), (1118, '2007-08-04', identity_val_local()),
      (1120, '2007-08-07', identity_val_local()), (2111, '2007-08-14', identity_val_local()),
      (2115, '2007-08-15', identity_val_local()), (2117, '2007-08-16', identity_val_local()),
      (2118, '2007-08-16', identity_val_local()), (3111, '2007-09-01', identity_val_local()),
      (3115, '2007-09-02', identity_val_local()), (3117, '2007-09-10', identity_val_local()),
      (3118, '2007-09-17', identity_val_local());

-- al doilea client:
INSERT INTO clienti (dencl, codfiscal, codpost, telefon)
      VALUES ('Client 2 SA', 'R1002', '700505', '0232212121') ;
-- inregistrarile copil al celui de-al doilea client:
INSERT INTO persclienti VALUES ('CNP2', identity_val_local(), 'Director general'),
      ('CNP3', identity_val_local(), 'Sef aprovizionare');
INSERT INTO facturi (NrFact, DataFact, CodCl) VALUES
(1113, '2007-08-01', identity_val_local()), (2113, '2007-08-14', identity_val_local()),
      (3113, '2007-09-02', identity_val_local());

-- al treilea client:
INSERT INTO clienti VALUES
      (DEFAULT, 'Client 3 SRL', 'R1003', 'Prosperitatii, 22', '706500','0235222222') ;
-- inregistrarile copil al celui de-al treilea client:
INSERT INTO persclienti VALUES ('CNP4', identity_val_local(), 'Sef aprovizionare'),
      ('CNP5', identity_val_local(), 'Director financiar');
INSERT INTO facturi (NrFact, DataFact, CodCl) VALUES
(1119, '2007-08-07', identity_val_local()), (2119, '2007-08-21', identity_val_local()),
 (3119, '2007-10-07', identity_val_local());

-- al patrulea client:
INSERT INTO clienti (dencl, codfiscal, adresa, codpost) VALUES
      ('Client 4', 'R1004', 'Sapientei, 56', '701150');
-- inregistrarile copil al celui de-al patrulea client:
INSERT INTO persclienti VALUES ('CNP6', identity_val_local(), 'Director general');
INSERT INTO facturi (NrFact, DataFact, CodCl) VALUES
      (1121, '2007-08-07', identity_val_local()), (2121, '2007-08-21', identity_val_local());

-- al cincilea client:
INSERT INTO clienti VALUES (DEFAULT, 'Client 5 SRL', 'R1005', NULL, '701900', '0256111111');
-- inregistrarile copil al celui de-al cincilea client:
INSERT INTO persclienti VALUES ('CNP7', identity_val_local(), 'Sef aprovizionare');
INSERT INTO facturi (nrfact, datafact, codcl, obs) VALUES
      (1112, '2007-08-01', identity_val_local(), 'Probleme cu transportul'),
      (1122, '2007-08-07', identity_val_local(), NULL),
      (2112, '2007-08-14', identity_val_local(),'Probleme cu transportul'),
      (2122, '2007-08-22', identity_val_local(), NULL),
      (3112, '2007-09-01', identity_val_local(), 'Probleme cu transportul');
```

```
-- al saselea client:
INSERT INTO clienti VALUES (DEFAULT, 'Client 6 SA', 'R1006', 'Pacientei, 33', '705550', NULL) ;
-- inregistrarile copil al celui de-al saselea client:
INSERT INTO persclienti VALUES ('CNP8', identity_val_local(), 'Director financiar');
INSERT INTO facturi (NrFact, DataFact, CodCl) VALUES
     (1114, '2007-08-01', identity_val_local());

-- al saptelea client:
INSERT INTO clienti VALUES
     (DEFAULT, 'Client 7 SRL', 'R1007', 'Victoria Capitalismului, 2', '701900', '0256121212') ;
-- inregistrarile copil al celui de-al saptelea client:
INSERT INTO persclienti VALUES ('CNP9', identity_val_local(), 'Sef aprovizionare');
INSERT INTO facturi (nrfact, datafact, codcl, obs) VALUES
     (1116, '2007-08-02', identity_val_local(), 'Pretul propus initial a fost modificat'),
     (2116, '2007-08-15', identity_val_local(), 'Pretul propus initial a fost modificat'),
     (3116, '2007-09-10', identity_val_local(), 'Pretul propus initial a fost modificat');

-- primul produs:
INSERT INTO produse VALUES (DEFAULT, 'Produs 1','buc', 'Tigari', .19) ;
-- liniile in care apare primul produs
INSERT INTO liniifact (nrfact, linie, codpr, cantitate, pretunit) VALUES
     (1111, 1, identity_val_local(), 50, 1000), (1117, 2, identity_val_local(), 100, 950),
     (1118, 2, identity_val_local(), 150, 930), (2111, 1, identity_val_local(), 57, 1000),
     (2117, 2, identity_val_local(), 110, 950), (2118, 2, identity_val_local(), 120, 930),
     (3111, 1, identity_val_local(), 57, 1000), (3117, 2, identity_val_local(), 110, 950),
     (3118, 2, identity_val_local(), 120, 930) ;

-- al doilea produs:
INSERT INTO produse VALUES (DEFAULT, 'Produs 2','kg', 'Bere', 0.09) ;
-- liniile in care apare al doilea produs
INSERT INTO liniifact (nrfact, linie, codpr, cantitate, pretunit) VALUES
     (1111, 2, identity_val_local(), 75, 1050), (1112, 1, identity_val_local(), 80, 1030),
     (1113, 1, identity_val_local(), 100, 975), (1114, 1, identity_val_local(), 70, 1070),
     (1115, 1, identity_val_local(), 150, 925), (1116, 1, identity_val_local(), 125, 930),
     (1117, 1, identity_val_local(), 100, 1000), (1118, 1, identity_val_local(), 30, 1100),
     (1119, 1, identity_val_local(), 35, 1090), (1120, 1, identity_val_local(), 80, 1120),
     (1121, 2, identity_val_local(), 100, 1050), (2111, 2, identity_val_local(), 79, 1050),
     (2112, 1, identity_val_local(), 85, 1030), (2113, 1, identity_val_local(), 120, 975),
     (2115, 1, identity_val_local(), 110, 925), (2116, 1, identity_val_local(), 135, 930),
     (2117, 1, identity_val_local(), 150, 1000), (2118, 1, identity_val_local(), 39, 1100),
     (2119, 1, identity_val_local(), 35, 1090), (2121, 2, identity_val_local(), 103, 1050),
     (3111, 2, identity_val_local(), 79, 1050), (3112, 1, identity_val_local(), 85, 1030),
     (3113, 1, identity_val_local(), 120, 975), (3115, 1, identity_val_local(), 110, 925),
     (3116, 1, identity_val_local(), 135, 930), (3117, 1, identity_val_local(), 150, 1000),
     (3118, 1, identity_val_local(), 39, 1100), (3119, 1, identity_val_local(), 35, 1090) ;

-- al treilea produs:
INSERT INTO produse VALUES (DEFAULT, 'Produs 3','kg', 'Bere', 0.19) ;
-- liniile in care apare al treilea produs
INSERT INTO liniifact (nrfact, linie, codpr, cantitate, pretunit) VALUES
     (1112, 2, identity_val_local(), 40, 750), (1119, 2, identity_val_local(), 40, 700),
     (2112, 2, identity_val_local(), 65, 750), (2119, 2, identity_val_local(), 40, 700),
     (3112, 2, identity_val_local(), 65, 750), (3119, 2, identity_val_local(), 40, 700) ;

-- al patrulea produs:
INSERT INTO produse VALUES (DEFAULT, 'Produs 4','l', 'Dulciuri', .09) ;
-- liniile in care apare al patrulea produs
INSERT INTO liniifact (nrfact, linie, codpr, cantitate, pretunit) VALUES
```

```
    (1114, 2, identity_val_local(), 30, 1705), (1119, 3, identity_val_local(), 50, 1410),
    (2119, 3, identity_val_local(), 55, 1410), (3119, 3, identity_val_local(), 55, 1410) ;

-- al cincilea produs:
INSERT INTO produse VALUES (DEFAULT, 'Produs 5','buc', 'Tigari', .19) ;
-- liniile in care apare al cincilea produs
INSERT INTO liniifact (nrfact, linie, codpr, cantitate, pretunit) VALUES
    (1111, 3, identity_val_local(), 500, 7060), (1114, 3, identity_val_local(), 700, 7064),
    (1119, 4, identity_val_local(), 750, 6300), (1121, 1, identity_val_local(), 550, 7064),
    (2111, 3, identity_val_local(), 510, 7060), (2119, 4, identity_val_local(), 755, 6300),
    (2121, 1, identity_val_local(), 550, 7064), (3111, 3, identity_val_local(), 510, 7060),
    (3119, 4, identity_val_local(), 755, 6300) ;

-- al saselea produs:
INSERT INTO produse VALUES (DEFAULT, 'Produs 6','p250g', 'Cafea', .19) ;
-- nu e nicio liniie in care sa apara al saselea produs

-- prima incasare
INSERT INTO incasari VALUES (DEFAULT, '2007-08-15', 'OP', '111', '2007-08-10' ) ;
-- facturile pentru prima incasare
INSERT INTO incasfact VALUES (identity_val_local(), 1111, 53996), (identity_val_local(), 1118,
101975) ;

-- a doua incasare
INSERT INTO incasari VALUES (DEFAULT, '2007-08-15', 'CHIT', '222', '2007-08-15') ;
-- facturile pentru a doua incasare
INSERT INTO incasfact VALUES (identity_val_local(), 1112, 125516) ;

-- a treia incasare
INSERT INTO incasari VALUES (DEFAULT, '2007-08-16', 'OP', '333', '2007-08-09') ;
-- facturile pentru a treia incasare
INSERT INTO incasfact VALUES (identity_val_local(), 1117, 9754),
    (identity_val_local(), 1118, 100000), (identity_val_local(), 1120, 7315) ;

-- a patra incasare
INSERT INTO incasari VALUES (DEFAULT, '2007-08-17', 'CEC', '444', '2007-08-10') ;
-- facturile pentru a patra incasare
INSERT INTO incasfact VALUES (identity_val_local(), 1117, 9754) ;

-- a cincea incasare
INSERT INTO incasari VALUES (DEFAULT, '2007-08-17', 'OP', '555', '2007-08-10') ;
-- facturile pentru a cincea incasare
INSERT INTO incasfact VALUES (identity_val_local(), 1113, 106275) ;

-- a sasea incasare
INSERT INTO incasari VALUES (DEFAULT, '2007-08-18', 'OP', '666', '2007-08-11') ;
-- facturile pentru a sasea incasare
INSERT INTO incasfact VALUES (identity_val_local(), 1117, 3696) ;

COMMIT ;
```

3.4.5. Scripturile de populare în Oracle

În Oracle inserările (listingurile 3.11 şi 3.12) sunt mult mai relaxate, grație inexistenței atributelor cu valori auto-generate. Constantele de tip dată calendaristice sunt prefixate, aşa cum am văzut de cuvântul DATE, iar cele de tip

moment de TIMESTAMP. Beneficiind de faptul că putem scrie mai multe comenzi pe aceeaşi linie, încercăm a mai scurta, cât de cât lungimea scripturilor, chiar dacă lizibilitatea scripturilor scade considerabil

Listing 3.11. Script Oracle de populare a tabelelor BD TRIAJ

```
DELETE FROM triaj ; DELETE FROM pacienti ; DELETE FROM  garzi ;
DELETE FROM  doctori ;

INSERT INTO doctori VALUES (1, 'Vasilcu Ionel', 'chirurgie', DATE'1965-11-11') ;
INSERT INTO doctori VALUES (2, 'Georgescu Mircea', 'hepatologie', DATE'1966-01-12') ;
INSERT INTO doctori VALUES (3, 'Zahir Tudorel', 'chirurgie', DATE'1965-11-11') ;
INSERT INTO doctori VALUES (4, 'Bostan Vasile', 'cardiologie', DATE'1965-11-11') ;
INSERT INTO doctori VALUES (5, 'Popescu Ionela', 'boli interne', DATE'1965-11-11') ;

INSERT INTO garzi VALUES
    (1, TIMESTAMP'2008-01-03 7:00:00', TIMESTAMP'2008-01-03 19:00:00') ;
INSERT INTO garzi VALUES
    (2, TIMESTAMP'2008-01-03 19:00:00', TIMESTAMP'2008-01-04 7:00:00') ;
INSERT INTO garzi VALUES
    (3, TIMESTAMP'2008-01-04 07:00:00',  TIMESTAMP'2008-01-04 18:00:00') ;
INSERT INTO garzi VALUES
    (4, TIMESTAMP'2008-01-04 18:00:00',  TIMESTAMP'2008-01-05 06:00:00') ;
INSERT INTO garzi VALUES
    (1, TIMESTAMP'2008-01-05 6:00:00', TIMESTAMP'2008-01-05 14:00:00') ;
INSERT INTO garzi VALUES
    (2, TIMESTAMP'2008-01-05 14:00:00', TIMESTAMP'2008-01-05 22:00:00') ;
INSERT INTO garzi VALUES
    (5, TIMESTAMP'2008-01-05 22:00:00',  TIMESTAMP'2008-01-06 9:00:00') ;

INSERT INTO pacienti (idpacient,numepacient, cnp, adresa, serie_nr_act_identitate)
    VALUES (1, 'Stroe Mihaela', 2601228390834, 'Bd. Cantemir, 32, Bl.G4, Sc.C, Ap.4', 'MR366766') ;
INSERT INTO pacienti (idpacient,numepacient, cnp, adresa, serie_nr_act_identitate)
    VALUES (2, 'Buzatu Corneliu', 1650512370514, 'Str. Desprimaveririi, 112', 'MX456783') ;
INSERT INTO pacienti (idpacient,numepacient, cnp, adresa, serie_nr_act_identitate)
    VALUES (3, 'Spineanu Marius', 5010101380625, 'Bd. Stefan cel Mare, 4, Bl.I1, Sc.A, Ap.24',
        'MX213345') ;
INSERT INTO pacienti (idpacient,numepacient, cnp, adresa, serie_nr_act_identitate)
    VALUES (4, 'Bagdasar Adela', 2601002250611, 'Str. Primaverii, 17', 'MX345678') ;
INSERT INTO pacienti (idpacient,numepacient, cnp, adresa, serie_nr_act_identitate)
    VALUES (5, 'Ionascu Ionelia', 6001122390199, 'Str. Florilor', 'MX654322') ;
INSERT INTO pacienti (idpacient,numepacient, cnp, adresa, serie_nr_act_identitate)
    VALUES (6, 'Popescu Maria-Mirabela', 2721231300888, 'Bd. 22 Decembrie, 2, Bl.5, Sc.B, Ap.21',
     'MX765432') ;
INSERT INTO pacienti (idpacient,numepacient, cnp, adresa, serie_nr_act_identitate)
    VALUES (7, 'Stroescu Mihaela Oana', 6020719120545, 'Str. Lenei Nr. 234', 'MX876567') ;
INSERT INTO pacienti (idpacient,numepacient, cnp, adresa, serie_nr_act_identitate)
    VALUES (8, 'Cazan Ana Maria', 2690202200345, 'Bd. Independentei, 87, Bl.K3, Sc.A, Ap.34',
     'MX987789') ;

INSERT INTO triaj VALUES (1, TIMESTAMP'2008-01-03 7:18:00', 1,
    'dureri de stomac intense', NULL, 'boli interne');
INSERT INTO triaj VALUES (2, TIMESTAMP'2008-01-03 8:45:00', 2,
    'febra puternica, varsaturi', 'penicilina', 'boli interne');
INSERT INTO triaj VALUES (3, TIMESTAMP'2008-01-03 12:45:00', 3,
    'deranjament stomacal', 'scobutil', NULL);
INSERT INTO triaj VALUES (4, TIMESTAMP'2008-01-03 20:45:00', 4,
    'palpitatii cardiace', 'linistin', 'cardiologie');
INSERT INTO triaj VALUES (5, TIMESTAMP'2008-01-04 1:28:00', 5,
```

```
    'plaga profunda picior drept', 'antibiotice, pansament', NULL);
INSERT INTO triaj VALUES (6, TIMESTAMP'2008-01-04 10:45:00', 3,
    'contractii stomacale, varsaturi', NULL, 'boli interne');
INSERT INTO triaj VALUES (7, TIMESTAMP'2008-01-04 11:20:00', 7,
    'fata de culoare galbena, ameteli', NULL, 'hepatologie');
INSERT INTO triaj VALUES (8, TIMESTAMP'2008-01-04 22:45:00', 8,
    'dureri articulare', 'scobutil', NULL);
INSERT INTO triaj VALUES (9, TIMESTAMP'2008-01-05 6:18:00', 5,
    'febra puternica, delir', 'penicilina', 'chirurgie');

COMMIT;
```

Din păcate, în Oracle nu putem enumera mai multe linii în clauza VALUES a comenzii INSERT, aşa încât listingul 3.12 are o lungime apreciabilă.

Listing 3.12. Script Oracle de populare a tabelelor BD VÎNZĂRI

```
DELETE FROM incasfact ; DELETE FROM incasari ; DELETE FROM liniifact ;
DELETE FROM facturi ; DELETE FROM produse ; DELETE FROM persclienti ;
DELETE FROM persoane ; DELETE FROM clienti ; DELETE FROM coduri_postale ;
DELETE FROM judete ;

INSERT INTO judete VALUES ('IS', 'Iasi', 'Moldova') ;
INSERT INTO judete VALUES ('VN', 'Vrancea', 'Moldova') ;
INSERT INTO judete VALUES ('NT', 'Neamt', 'Moldova') ;
INSERT INTO judete VALUES ('SV', 'Suceava', 'Moldova') ;
INSERT INTO judete VALUES ('VS', 'Vaslui', 'Moldova') ;
INSERT INTO judete VALUES ('TM', 'Timis', 'Banat') ;

INSERT INTO coduri_postale VALUES ('700505', 'Iasi', 'IS') ;
INSERT INTO coduri_postale VALUES ('700510', 'Iasi', 'IS') ;
INSERT INTO coduri_postale VALUES ('700515', 'Iasi', 'IS') ;
INSERT INTO coduri_postale VALUES ('701150', 'Pascani', 'IS') ;
INSERT INTO coduri_postale VALUES ('706500', 'Vaslui', 'VS') ;
INSERT INTO coduri_postale VALUES ('706510', 'Vaslui', 'VS') ;
INSERT INTO coduri_postale VALUES ('705300', 'Focsani', 'VN') ;
INSERT INTO coduri_postale VALUES ('705310', 'Focsani', 'VN') ;
INSERT INTO coduri_postale VALUES ('706400', 'Birlad', 'VS') ;
INSERT INTO coduri_postale VALUES ('705800', 'Suceava', 'SV') ;
INSERT INTO coduri_postale VALUES ('705550', 'Roman', 'NT') ;
INSERT INTO coduri_postale VALUES ('701900', 'Timisoara', 'TM') ;

INSERT INTO clienti VALUES (1001, 'Client 1 SRL', 'R1001', 'Tranzitiei, 13 bis',
    '700505', NULL) ;
INSERT INTO clienti (codcl, dencl, codfiscal, codpost, telefon)
    VALUES (1002,'Client 2 SA', 'R1002', '700505', '0232212121') ;
INSERT INTO clienti VALUES (1003, 'Client 3 SRL', 'R1003', 'Prosperitatii, 22',
    '706500','0235222222') ;
INSERT INTO clienti (codcl, dencl, adresa, codfiscal, codpost)
    VALUES (1004, 'Client 4', 'Sapientei, 56', 'R1004', '701150');
INSERT INTO clienti VALUES (1005, 'Client 5 SRL', 'R1005', NULL,
    '701900', '0256111111');
INSERT INTO clienti VALUES (1006, 'Client 6 SA', 'R1006', 'Pacientei, 33',
    '705550', NULL) ;
INSERT INTO clienti VALUES (1007, 'Client 7 SRL', 'R1007', 'Victoria Capitalismului, 2',
    '701900', '0256121212') ;

INSERT INTO persoane VALUES ('CNP1', 'Ioan', 'Vasile', 'I.L.Caragiale, 22',       'B',
```

```
    '700505', '123456', '987654', '094222222', NULL) ;
INSERT INTO persoane VALUES ('CNP2', 'Vasile', 'Ion', NULL,          'B',
    '700505', '234567', '876543', '094222223', 'Ion@a.ro') ;
INSERT INTO persoane VALUES ('CNP3', 'Popovici', 'Ioana', 'V.Micle, Bl.I, Sc.B,Ap.2', 'F',
    '701150', '345678', NULL, '094222224', NULL) ;
INSERT INTO persoane VALUES ('CNP4', 'Lazar', 'Caraion', 'M.Eminescu, 42', 'B',
    '706500', '456789', NULL, '094222225', NULL) ;
INSERT INTO persoane VALUES ('CNP5', 'Iurea', 'Simion', 'I.Creanga, 44 bis', 'B',
     '706500', '567890', '543210', NULL, NULL) ;
INSERT INTO persoane VALUES ('CNP6', 'Vasc', 'Simona', 'M.Eminescu, 13', 'F',
    '701150', NULL, '432109', '094222227', NULL) ;
INSERT INTO persoane VALUES ('CNP7', 'Popa', 'Ioanid', 'I.Ion, Bl.H2, Sc.C, Ap.45', 'B',
    '701900', '789012', '321098', NULL, NULL) ;
INSERT INTO persoane VALUES ('CNP8', 'Bogacs', 'Ildiko', 'I.V.Viteazu, 67', 'F',
    '705550', '890123', '210987', '094222229', NULL) ;
INSERT INTO persoane VALUES ('CNP9', 'Ioan', 'Vasilica', 'Garii, Bl.B4, Sc.A, Ap.1', 'F',
    '701900', '901234', '109876', '094222230', NULL) ;

INSERT INTO persclienti VALUES ('CNP1', 1001, 'Director general');
INSERT INTO persclienti VALUES ('CNP2', 1002, 'Director general');
INSERT INTO persclienti VALUES ('CNP3', 1002, 'Sef aprovizionare');
INSERT INTO persclienti VALUES ('CNP4', 1003, 'Sef aprovizionare');
INSERT INTO persclienti VALUES ('CNP5', 1003, 'Director financiar');
INSERT INTO persclienti VALUES ('CNP6', 1004, 'Director general');
INSERT INTO persclienti VALUES ('CNP7', 1005, 'Sef aprovizionare');
INSERT INTO persclienti VALUES ('CNP8', 1006, 'Director financiar');
INSERT INTO persclienti VALUES ('CNP9', 1007, 'Sef aprovizionare');

INSERT INTO produse VALUES (1, 'Produs 1','buc', 'Tigari', .19) ;
INSERT INTO produse VALUES (2, 'Produs 2','kg', 'Bere', 0.09) ;
INSERT INTO produse VALUES (3, 'Produs 3','kg', 'Bere', 0.19) ;
INSERT INTO produse VALUES (4, 'Produs 4','l', 'Dulciuri', .09) ;
INSERT INTO produse VALUES (5, 'Produs 5','buc', 'Tigari', .19) ;
INSERT INTO produse VALUES (6, 'Produs 6','p250g', 'Cafea', .19) ;

INSERT INTO facturi (NrFact, DataFact, CodCl) VALUES (1111, DATE'2007-08-01', 1001);
INSERT INTO facturi (nrfact, datafact, codcl, obs) VALUES (1112, DATE'2007-08-01', 1005,
     'Probleme cu transportul');
INSERT INTO facturi (NrFact, DataFact, CodCl) VALUES (1113, DATE'2007-08-01', 1002);
INSERT INTO facturi (NrFact, DataFact, CodCl) VALUES (1114, DATE'2007-08-01', 1006);
INSERT INTO facturi (NrFact, DataFact, CodCl) VALUES (1115, DATE'2007-08-02', 1001);
INSERT INTO facturi (nrfact, datafact, codcl, obs) VALUES (1116, DATE'2007-08-02', 1007,
    'Pretul propus initial a fost modificat');
INSERT INTO facturi (NrFact, DataFact, CodCl) VALUES (1117, DATE'2007-08-03', 1001);
INSERT INTO facturi (NrFact, DataFact, CodCl) VALUES (1118, DATE'2007-08-04', 1001);
INSERT INTO facturi (NrFact, DataFact, CodCl) VALUES (1119, DATE'2007-08-07', 1003);
INSERT INTO facturi (NrFact, DataFact, CodCl) VALUES (1120, DATE'2007-08-07', 1001);
INSERT INTO facturi (NrFact, DataFact, CodCl) VALUES (1121, DATE'2007-08-07', 1004);
INSERT INTO facturi (NrFact, DataFact, CodCl) VALUES (1122, DATE'2007-08-07', 1005);
INSERT INTO facturi (NrFact, DataFact, CodCl) VALUES (2111, DATE'2007-08-14', 1001);
INSERT INTO facturi (nrfact, datafact, codcl, obs)
    VALUES (2112, DATE'2007-08-14', 1005,  'Probleme cu transportul');
INSERT INTO facturi (NrFact, DataFact, CodCl) VALUES (2113, DATE'2007-08-14', 1002);
INSERT INTO facturi (NrFact, DataFact, CodCl) VALUES (2115, DATE'2007-08-15', 1001);
INSERT INTO facturi (nrfact, datafact, codcl, obs)
    VALUES (2116, DATE'2007-08-15', 1007, 'Pretul propus initial a fost modificat');
INSERT INTO facturi (NrFact, DataFact, CodCl) VALUES (2117, DATE'2007-08-16', 1001);
INSERT INTO facturi (NrFact, DataFact, CodCl) VALUES (2118, DATE'2007-08-16', 1001);
INSERT INTO facturi (NrFact, DataFact, CodCl) VALUES (2119, DATE'2007-08-21', 1003);
```

```
INSERT INTO facturi (NrFact, DataFact, CodCl) VALUES (2121, DATE'2007-08-21', 1004);
INSERT INTO facturi (NrFact, DataFact, CodCl) VALUES (2122, DATE'2007-08-22', 1005);
INSERT INTO facturi (NrFact, DataFact, CodCl) VALUES (3111, DATE'2007-09-01', 1001);
INSERT INTO facturi (nrfact, datafact, codcl, obs)
    VALUES (3112, DATE'2007-09-01', 1005, 'Probleme cu transportul');
INSERT INTO facturi (NrFact, DataFact, CodCl) VALUES (3113, DATE'2007-09-02', 1002);
INSERT INTO facturi (NrFact, DataFact, CodCl) VALUES (3115, DATE'2007-09-02', 1001);
INSERT INTO facturi (nrfact, datafact, codcl, obs)
    VALUES (3116, DATE'2007-09-10', 1007,  'Pretul propus initial a fost modificat');
INSERT INTO facturi (NrFact, DataFact, CodCl) VALUES (3117, DATE'2007-09-10', 1001);
INSERT INTO facturi (NrFact, DataFact, CodCl) VALUES (3118, DATE'2007-09-17', 1001);
INSERT INTO facturi (NrFact, DataFact, CodCl) VALUES (3119, DATE'2007-10-07', 1003);

INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (1111,1,1,50,1000) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (1111,2,2,75,1050) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (1111,3,5,500,7060) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (1112,1,2,80,1030) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (1112,2,3,40,750) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (1113,1,2,100,975) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (1114,1,2,70,1070) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (1114,2,4,30,1705) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (1114,3,5,700,7064) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (1115,1,2,150,925) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (1116,1,2,125,930) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (1117,1,2,100,1000) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (1117,2,1,100,950) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (1118,1,2,30,1100) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (1118,2,1,150,930) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (1119,1,2,35,1090) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (1119,2,3,40,700) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (1119,3,4,50,1410) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (1119,4,5,750,6300) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (1120,1,2,80,1120) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (1121,1,5,550,7064) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (1121,2,2,100,1050) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (2111,1,1,57,1000) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (2111,2,2,79,1050) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (2111,3,5,510,7060) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (2112,1,2,85,1030) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (2112,2,3,65,750) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (2113,1,2,120,975) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (2115,1,2,110,925) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (2116,1,2,135,930) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (2117,1,2,150,1000) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (2117,2,1,110,950) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (2118,1,2,39,1100) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (2118,2,1,120,930) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (2119,1,2,35,1090) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (2119,2,3,40,700) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (2119,3,4,55,1410) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (2119,4,5,755,6300) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (2121,1,5,550,7064) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (2121,2,2,103,1050) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (3111,1,1,57,1000) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (3111,2,2,79,1050) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (3111,3,5,510,7060) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (3112,1,2,85,1030) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (3112,2,3,65,750) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (3113,1,2,120,975) ;
```

```
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (3115,1,2,110,925) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (3116,1,2,135,930) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (3117,1,2,150,1000) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (3117,2,1,110,950) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (3118,1,2,39,1100) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (3118,2,1,120,930) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (3119,1,2,35,1090) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (3119,2,3,40,700) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (3119,3,4,55,1410) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (3119,4,5,755,6300) ;

INSERT INTO incasari VALUES (1234, DATE'2007-08-15', 'OP', '111', DATE'2007-08-10' ) ;
INSERT INTO incasari VALUES (1235, DATE'2007-08-15','CHIT', '222', DATE'2007-08-15') ;
INSERT INTO incasari VALUES (1236, DATE'2007-08-16', 'OP', '333', DATE'2007-08-09') ;
INSERT INTO incasari VALUES (1237, DATE'2007-08-17', 'CEC', '444', DATE'2007-08-10') ;
INSERT INTO incasari VALUES (1238, DATE'2007-08-17', 'OP', '555', DATE'2007-08-10') ;
INSERT INTO incasari VALUES (1239, DATE'2007-08-18', 'OP', '666', DATE'2007-08-11') ;

INSERT INTO incasfact VALUES (1234, 1111, 53996) ;
INSERT INTO incasfact VALUES (1234, 1118, 101975) ;
INSERT INTO incasfact VALUES (1235, 1112, 125516) ;
INSERT INTO incasfact VALUES (1236, 1117, 9754) ;
INSERT INTO incasfact VALUES (1236, 1118, 100000) ;
INSERT INTO incasfact VALUES (1236, 1120, 7315) ;
INSERT INTO incasfact VALUES (1237, 1117, 9754) ;
INSERT INTO incasfact VALUES (1238, 1113, 106275) ;
INSERT INTO incasfact VALUES (1239, 1117, 3696) ;

COMMIT ;
```

3.4.6. Scripturile de populare în PostgreSQL

PostgreSQL-ul este de-a dreptul interesant. Vă amintiți (listingul 3.5) că, pentru atributele IdDoctor din tabela DOCTORI şi IdPacient din tabela PACIENȚI, am recurs la tipul SERIAL, iar pentru IdExaminare din tabela TRIAJ la tipul BIGSERIAL[36]. Ne-am aştepta, astfel, ca lansarea scriptului Oracle (listing 3.11) în PostgreSQL să se soldeze cu problemele întâlnite în DB2 pentru coloanele autogenerate, întrucât comenzile INSERT conțin valori pentru attributele SERIAL/BIGSERIAL. Ei bine, PostgreSQL-ul nu are nicio obiecție ! Mai mult, nici dacă lansăm scriptul din listingul 3.11 de mai multe ori, nu recepționăm nicio eroare. Explicația stă în faptul că pentru PostgreSQL, valorile "seriale" depind decisiv de liniile existente în tabelă, iar o valoare specificată în comanda INSERT

[36] Nici SERIAL, nici BIGSERIAL nu sunt tipuri propriu-zise.

suprascrie valoarea implicită. Deşi pare o binecuvântare, această relaxare poate genera însă probleme destul de neplăcute, cum ar fi violarea ulterioară a cheii primare, prin suprapunerea următoarei valori autogenerate cu o valoare anterior introdusă printr-o comandă INSERT. Este motivul pentru care în tabelele BD VÎNZĂRI nu am mai recurs la tipurile SERIAL şi BIGSERIAL.

Listingul de populare a acestor tabele este identic cu cel din Oracle (3.12), chiar dacă, similar DB2, PostgreSQL permite inserarea mai multor înregistrări cu o singură comandă INSERT:

```
INSERT INTO judete VALUES
        ('IS', 'Iasi', 'Moldova'), ('VN', 'Vrancea', 'Moldova'), ('NT', 'Neamt', 'Moldova'),
        ('SV', 'Suceava', 'Moldova'), ('VS', 'Vaslui', 'Moldova'), ('TM', 'Timis', 'Banat') ;
```

3.4.7. Scripturile de populare în SQL Server

SQL Server este, ca şi DB2, scrupulos în privința atributelor IDENTITY. Dacă în DB2 funcționează varianta *INSERT INTO doctori VALUES (DEFAULT, 'Vasilcu Ionel', 'chirurgie', '1965-11-11')* , în SQL Server nu, sintaxa acceptată fiind:

```
INSERT INTO doctori (numedoctor, specialitate, datanasterii)
    VALUES ('Vasilcu Ionel', 'chirurgie',  '1965-11-11') ;
```

De asemenea, execuții repetate ale scriptului de populare care începe cu golirea tabelelor (DELETE FROM ...) consumă valorile autoincrementate, astfel încât apar probleme cu specificarea valorilor pentru cheile străine. Scriptul de populare SQL Server a tabelelor din BD TRIAJ este prezentat în listing 3.13.

Listing 3.13. Script MS SQL Server de populare a tabelelor BD TRIAJ

```
DELETE FROM triaj ; DELETE FROM pacienti ;
DELETE FROM  garzi ; DELETE FROM  doctori ;

INSERT INTO doctori (numedoctor, specialitate,datanasterii)
    VALUES ('Vasilcu Ionel', 'chirurgie',  '1965-11-11') ;
INSERT INTO doctori (numedoctor, specialitate,datanasterii)
    VALUES ('Georgescu Mircea', 'hepatologie',  '1966-01-12') ;
INSERT INTO doctori (numedoctor, specialitate,datanasterii)
    VALUES ('Zahir Tudorel', 'chirurgie',  '1965-11-11') ;
INSERT INTO doctori (numedoctor, specialitate,datanasterii)
    VALUES ('Bostan Vasile', 'cardiologie',  '1965-11-11') ;
INSERT INTO doctori (numedoctor, specialitate,datanasterii)
    VALUES ('Popescu Ionela', 'boli interne',  '1965-11-11') ;

INSERT INTO garzi VALUES (1,  '2008-01-03 7:00:00',  '2008-01-03 19:00:00') ;
INSERT INTO garzi VALUES (2,  '2008-01-03 19:00:00',  '2008-01-04 7:00:00') ;
INSERT INTO garzi VALUES (3,  '2008-01-04 07:00:00',  '2008-01-04 18:00:00') ;
INSERT INTO garzi VALUES (4,  '2008-01-04 18:00:00',  '2008-01-05 06:00:00') ;
INSERT INTO garzi VALUES (1,  '2008-01-05 6:00:00',  '2008-01-05 14:00:00') ;
INSERT INTO garzi VALUES (2,  '2008-01-05 14:00:00',  '2008-01-05 22:00:00') ;
INSERT INTO garzi VALUES (5,  '2008-01-05 22:00:00',  '2008-01-06 9:00:00') ;

INSERT INTO pacienti (numepacient, CNP, adresa, serie_nr_act_identitate)
    VALUES ('Stroe Mihaela', 2601228390834, 'Bd. Cantemir, 32, Bl.G4, Sc.C, Ap.4', 'MR366766') ;
INSERT INTO pacienti (numepacient, CNP, adresa, serie_nr_act_identitate)
```

```
        VALUES ('Buzatu Corneliu', 1650512370514, 'Str. Desprimaveririi, 112', 'MX456783') ;
    INSERT INTO pacienti (numepacient, CNP, adresa, serie_nr_act_identitate)
        VALUES ('Spineanu Marius', 5010101380625, 'Bd. Stefan cel Mare, 4, Bl.I1, Sc.A, Ap.24',
            'MX213345') ;
    INSERT INTO pacienti (numepacient, CNP, adresa, serie_nr_act_identitate)
        VALUES ('Bagdasar Adela', 2601002250611, 'Str. Primaverii, 17', 'MX345678') ;
    INSERT INTO pacienti (numepacient, CNP, adresa, serie_nr_act_identitate)
        VALUES ('Ionascu Ionelia', 6001122390199, 'Str. Florilor', 'MX654322') ;
    INSERT INTO pacienti (numepacient, CNP, adresa, serie_nr_act_identitate)
        VALUES ('Popescu Maria-Mirabela', 2721231300888, 'Bd. 22 Decembrie, 2, Bl.5, Sc.B, Ap.21',
            'MX765432') ;
    INSERT INTO pacienti (numepacient, CNP, adresa, serie_nr_act_identitate)
        VALUES ('Stroescu Mihaela Oana', 6020719120545, 'Str. Lenei Nr. 234', 'MX876567') ;
    INSERT INTO pacienti (numepacient, CNP, adresa, serie_nr_act_identitate)
        VALUES ('Cazan Ana Maria', 2690202200345, 'Bd. Independentei, 87, Bl.K3, Sc.A, Ap.34',
            'MX987789') ;

INSERT INTO triaj (dataora_examinare, idpacient, simptome, tratament_imediat, sectie_destinatie)
    VALUES ('2008-01-03 7:18:00', 1, 'dureri de stomac intense', NULL, 'boli interne');
INSERT INTO triaj (dataora_examinare, idpacient, simptome, tratament_imediat, sectie_destinatie)
    VALUES ('2008-01-03 8:45:00', 2, 'febra puternica, varsaturi', 'penicilina', 'boli interne');
INSERT INTO triaj (dataora_examinare, idpacient, simptome, tratament_imediat, sectie_destinatie)
    VALUES ('2008-01-03 12:45:00', 3, 'deranjament stomacal', 'scobutil', NULL);
INSERT INTO triaj (dataora_examinare, idpacient, simptome, tratament_imediat, sectie_destinatie)
    VALUES ('2008-01-03 20:45:00', 4, 'palpitatii cardiace', 'linistin', 'cardiologie');
INSERT INTO triaj (dataora_examinare, idpacient, simptome, tratament_imediat, sectie_destinatie)
    VALUES ('2008-01-04 1:28:00', 5, 'plaga profunda picior drept', 'antibiotice, pansament', NULL);
INSERT INTO triaj (dataora_examinare, idpacient, simptome, tratament_imediat, sectie_destinatie)
    VALUES ('2008-01-04 10:45:00', 3, 'contractii stomacale, varsaturi', NULL, 'boli interne');
INSERT INTO triaj (dataora_examinare, idpacient, simptome, tratament_imediat, sectie_destinatie)
    VALUES ('2008-01-04 11:20:00', 7, 'fata de culoare galbena, ameteli', NULL, 'hepatologie');
INSERT INTO triaj (dataora_examinare, idpacient, simptome, tratament_imediat, sectie_destinatie)
    VALUES ('2008-01-04 22:45:00', 8, 'dureri articulare', 'scobutil', NULL);
INSERT INTO triaj (dataora_examinare, idpacient, simptome, tratament_imediat, sectie_destinatie)
    VALUES ('2008-01-05 6:18:00', 5, 'febra puternica, delir', 'penicilina', 'chirurgie');
```

În privința comenzilor de populare a tabelelor din BD VÎNZĂRI, nu ar fi prea multe de adăugat – vezi listing 3.14 – doar că pot fi grupate mai multe comenzi pe aceeaşi linie a scriptului, însă comanda INSERT nu permite adăugarea simultană a două sau mai multe linii într-o tabelă.

Listing 3.14. Script MS SQL Server de populare a tabelelor BD VÎNZĂRI

```
DELETE FROM incasfact ; DELETE FROM incasari ; DELETE FROM liniifact ;
DELETE FROM facturi ; DELETE FROM produse ;DELETE FROM persclienti ;
DELETE FROM persoane ; DELETE FROM clienti ; DELETE FROM coduri_postale ;
DELETE FROM judete ;

INSERT INTO judete VALUES ('IS', 'Iasi', 'Moldova') ;
INSERT INTO judete VALUES ('VN', 'Vrancea', 'Moldova') ;
INSERT INTO judete VALUES ('NT', 'Neamt', 'Moldova') ;
INSERT INTO judete VALUES ('SV', 'Suceava', 'Moldova') ;
INSERT INTO judete VALUES ('VS', 'Vaslui', 'Moldova') ;
INSERT INTO judete VALUES ('TM', 'Timis', 'Banat') ;

INSERT INTO coduri_postale VALUES ('700505', 'Iasi', 'IS') ;
INSERT INTO coduri_postale VALUES ('700510', 'Iasi', 'IS') ;
INSERT INTO coduri_postale VALUES ('700515', 'Iasi', 'IS') ;
```

```
INSERT INTO coduri_postale VALUES ('701150', 'Pascani', 'IS') ;
INSERT INTO coduri_postale VALUES ('706500', 'Vaslui', 'VS') ;
INSERT INTO coduri_postale VALUES ('706510', 'Vaslui', 'VS') ;
INSERT INTO coduri_postale VALUES ('705300', 'Focsani', 'VN') ;
INSERT INTO coduri_postale VALUES ('705310', 'Focsani', 'VN') ;
INSERT INTO coduri_postale VALUES ('706400', 'Birlad', 'VS') ;
INSERT INTO coduri_postale VALUES ('705800', 'Suceava', 'SV') ;
INSERT INTO coduri_postale VALUES ('705550', 'Roman', 'NT') ;
INSERT INTO coduri_postale VALUES ('701900', 'Timisoara', 'TM') ;

INSERT INTO clienti (DenCl, CodFiscal, Adresa, CodPost, Telefon)
    VALUES ('Client 1 SRL', 'R1001', 'Tranzitiei, 13 bis', '700505', NULL) ;
INSERT INTO clienti (DenCl, CodFiscal, CodPost, Telefon)
    VALUES ('Client 2 SA', 'R1002', '700505', '0232212121') ;
INSERT INTO clienti (DenCl, CodFiscal, Adresa, CodPost, Telefon)
     VALUES ('Client 3 SRL', 'R1003', 'Prosperitatii, 22', '706500','0235222222') ;
INSERT INTO clienti (DenCl, CodFiscal, Adresa, CodPost)
    VALUES ('Client 4', 'R1004', 'Sapientei, 56', '701150');
INSERT INTO clienti (DenCl, CodFiscal, Adresa, CodPost, Telefon)
    VALUES ('Client 5 SRL', 'R1005', NULL, '701900', '0256111111');
INSERT INTO clienti  (DenCl, CodFiscal, Adresa, CodPost, Telefon)
    VALUES ('Client 6 SA', 'R1006', 'Pacientei, 33', '705550', NULL) ;
INSERT INTO clienti (DenCl, CodFiscal, Adresa, CodPost, Telefon) VALUES
('Client 7 SRL', 'R1007', 'Victoria Capitalismului, 2', '701900', '0256121212') ;

INSERT INTO persoane VALUES ('CNP1', 'Ioan', 'Vasile', 'I.L.Caragiale, 22', 'B', '700505', '123456',
     '987654', '094222222', NULL) ;
INSERT INTO persoane VALUES ('CNP2', 'Vasile', 'Ion', NULL, 'B', '700505', '234567', '876543',
     '094222223', 'Ion@a.ro') ;
INSERT INTO persoane VALUES ('CNP3', 'Popovici', 'Ioana', 'V.Micle, Bl.I, Sc.B,Ap.2', 'F', '701150',
     '345678', NULL, '094222224', NULL) ;
INSERT INTO persoane VALUES ('CNP4', 'Lazar', 'Caraion', 'M.Eminescu, 42', 'B', '706500',
'456789',
     NULL, '094222225', NULL) ;
INSERT INTO persoane VALUES ('CNP5', 'Iurea', 'Simion', 'I.Creanga, 44 bis', 'B', '706500',
'567890',
     '543210', NULL, NULL) ;
INSERT INTO persoane VALUES ('CNP6', 'Vasc', 'Simona', 'M.Eminescu, 13', 'F', '701150', NULL,
     '432109', '094222227', NULL) ;
INSERT INTO persoane VALUES ('CNP7', 'Popa', 'Ioanid', 'I.Ion, Bl.H2, Sc.C, Ap.45', 'B', '701900',
     '789012', '321098', NULL, NULL) ;
INSERT INTO persoane VALUES ('CNP8', 'Bogacs', 'Ildiko', 'I.V.Viteazu, 67', 'F', '705550', '890123',
     '210987', '094222229', NULL) ;
INSERT INTO persoane VALUES ('CNP9', 'Ioan', 'Vasilica', 'Garii, Bl.B4, Sc.A, Ap.1', 'F', '701900',
     '901234', '109876', '094222230', NULL) ;

INSERT INTO persclienti VALUES ('CNP1', 1001, 'Director general');
INSERT INTO persclienti VALUES ('CNP2', 1002, 'Director general');
INSERT INTO persclienti VALUES ('CNP3', 1002, 'Sef aprovizionare');
INSERT INTO persclienti VALUES ('CNP4', 1003, 'Sef aprovizionare');
INSERT INTO persclienti VALUES ('CNP5', 1003, 'Director financiar');
INSERT INTO persclienti VALUES ('CNP6', 1004, 'Director general');
INSERT INTO persclienti VALUES ('CNP7', 1005, 'Sef aprovizionare');
INSERT INTO persclienti VALUES ('CNP8', 1006, 'Director financiar');
INSERT INTO persclienti VALUES ('CNP9', 1007, 'Sef aprovizionare');

INSERT INTO produse VALUES ('Produs 1','buc', 'Tigari', .19) ;
INSERT INTO produse VALUES ('Produs 2','kg', 'Bere', 0.09) ;
INSERT INTO produse VALUES ('Produs 3','kg', 'Bere', 0.19) ;
```

```
INSERT INTO produse VALUES ('Produs 4','l', 'Dulciuri', .09) ;
INSERT INTO produse VALUES ('Produs 5','buc', 'Tigari', .19) ;
INSERT INTO produse VALUES ('Produs 6','p250g', 'Cafea', .19) ;

INSERT INTO facturi (NrFact, DataFact, CodCl) VALUES (1111, '2007-08-01', 1001);
INSERT INTO facturi (NrFact, DataFact, CodCl, Obs) VALUES (1112, '2007-08-01', 1005,
     'Probleme cu transportul');
INSERT INTO facturi (NrFact, DataFact, CodCl) VALUES (1113, '2007-08-01', 1002);
INSERT INTO facturi (NrFact, DataFact, CodCl) VALUES (1114, '2007-08-01', 1006);
INSERT INTO facturi (NrFact, DataFact, CodCl) VALUES (1115, '2007-08-02', 1001);
INSERT INTO facturi (NrFact, DataFact, CodCl, Obs) VALUES (1116, '2007-08-02', 1007,
    'Pretul propus initial a fost modificat');
INSERT INTO facturi (NrFact, DataFact, CodCl) VALUES (1117, '2007-08-03', 1001);
INSERT INTO facturi (NrFact, DataFact, CodCl) VALUES (1118, '2007-08-04', 1001);
INSERT INTO facturi (NrFact, DataFact, CodCl) VALUES (1119, '2007-08-07', 1003);
INSERT INTO facturi (NrFact, DataFact, CodCl) VALUES (1120, '2007-08-07', 1001);
INSERT INTO facturi (NrFact, DataFact, CodCl) VALUES (1121, '2007-08-07', 1004);
INSERT INTO facturi (NrFact, DataFact, CodCl) VALUES (1122, '2007-08-07', 1005);
INSERT INTO facturi (NrFact, DataFact, CodCl) VALUES (2111, '2007-08-14', 1001);
INSERT INTO facturi (NrFact, DataFact, CodCl, Obs) VALUES (2112, '2007-08-14', 1005,
    'Probleme cu transportul');
INSERT INTO facturi (NrFact, DataFact, CodCl) VALUES (2113, '2007-08-14', 1002);
INSERT INTO facturi (NrFact, DataFact, CodCl) VALUES (2115, '2007-08-15', 1001);
INSERT INTO facturi (NrFact, DataFact, CodCl, Obs) VALUES (2116, '2007-08-15', 1007,
    'Pretul propus initial a fost modificat');
INSERT INTO facturi (NrFact, DataFact, CodCl) VALUES (2117, '2007-08-16', 1001);
INSERT INTO facturi (NrFact, DataFact, CodCl) VALUES (2118, '2007-08-16', 1001);
INSERT INTO facturi (NrFact, DataFact, CodCl) VALUES (2119, '2007-08-21', 1003);
INSERT INTO facturi (NrFact, DataFact, CodCl) VALUES (2121, '2007-08-21', 1004);
INSERT INTO facturi (NrFact, DataFact, CodCl)  VALUES (2122, '2007-08-22', 1005);
INSERT INTO facturi (NrFact, DataFact, CodCl) VALUES (3111, '2007-09-01', 1001);
INSERT INTO facturi (NrFact, DataFact, CodCl, Obs) VALUES (3112, '2007-09-01', 1005,
    'Probleme cu transportul');
INSERT INTO facturi (NrFact, DataFact, CodCl) VALUES (3113, '2007-09-02', 1002);
INSERT INTO facturi (NrFact, DataFact, CodCl) VALUES (3115, '2007-09-02', 1001);
INSERT INTO facturi (NrFact, DataFact, CodCl, Obs)  VALUES (3116, '2007-09-10', 1007,
     'Pretul propus initial a fost modificat');
INSERT INTO facturi (NrFact, DataFact, CodCl) VALUES (3117, '2007-09-10', 1001);
INSERT INTO facturi (NrFact, DataFact, CodCl) VALUES (3118, '2007-09-17', 1001);
INSERT INTO facturi (NrFact, DataFact, CodCl) VALUES (3119, '2007-10-07', 1003);

INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit) VALUES (1111,1,1,50,1000) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (1111,2,2,75,1050) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (1111,3,5,500,7060) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (1112,1,2,80,1030) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (1112,2,3,40,750) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (1113,1,2,100,975) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (1114,1,2,70,1070) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (1114,2,4,30,1705) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (1114,3,5,700,7064) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (1115,1,2,150,925) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (1116,1,2,125,930) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (1117,1,2,100,1000) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (1117,2,1,100,950) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (1118,1,2,30,1100) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (1118,2,1,150,930) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (1119,1,2,35,1090) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (1119,2,3,40,700) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (1119,3,4,50,1410) ;
```

```
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (1119,4,5,750,6300) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (1120,1,2,80,1120) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (1121,1,5,550,7064) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (1121,2,2,100,1050) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (2111,1,1,57,1000) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (2111,2,2,79,1050) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (2111,3,5,510,7060) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (2112,1,2,85,1030) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (2112,2,3,65,750) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (2113,1,2,120,975) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (2115,1,2,110,925) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (2116,1,2,135,930) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (2117,1,2,150,1000) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (2117,2,1,110,950) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (2118,1,2,39,1100) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (2118,2,1,120,930) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (2119,1,2,35,1090) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (2119,2,3,40,700) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (2119,3,4,55,1410) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (2119,4,5,755,6300) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (2121,1,5,550,7064) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (2121,2,2,103,1050) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (3111,1,1,57,1000) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (3111,2,2,79,1050) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (3111,3,5,510,7060) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (3112,1,2,85,1030) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (3112,2,3,65,750) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (3113,1,2,120,975) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (3115,1,2,110,925) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (3116,1,2,135,930) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (3117,1,2,150,1000) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (3117,2,1,110,950) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (3118,1,2,39,1100) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (3118,2,1,120,930) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (3119,1,2,35,1090) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (3119,2,3,40,700) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (3119,3,4,55,1410) ;
INSERT INTO liniifact (NrFact,Linie,CodPr,Cantitate,PretUnit)  VALUES (3119,4,5,755,6300) ;

INSERT INTO incasari (DataInc,CodDoc,NrDoc, DataDoc) VALUES
    ('2007-08-15', 'OP', '111', '2007-08-10' ) ;
INSERT INTO incasari (DataInc,CodDoc,NrDoc, DataDoc) VALUES
    ('2007-08-15', 'CHIT', '222', '2007-08-15') ;
INSERT INTO incasari (DataInc,CodDoc,NrDoc, DataDoc) VALUES
    ('2007-08-16', 'OP', '333', '2007-08-09') ;
INSERT INTO incasari (DataInc,CodDoc,NrDoc, DataDoc) VALUES
     ('2007-08-17', 'CEC', '444', '2007-08-10') ;
INSERT INTO incasari (DataInc,CodDoc,NrDoc, DataDoc) VALUES
     ('2007-08-17', 'OP', '555', '2007-08-10') ;
INSERT INTO incasari (DataInc,CodDoc,NrDoc, DataDoc) VALUES
     ('2007-08-18', 'OP', '666', '2007-08-11') ;

INSERT INTO incasfact VALUES (1234, 1111, 53996) ;
INSERT INTO incasfact VALUES (1234, 1118, 101975) ;
INSERT INTO incasfact VALUES (1235, 1112, 125516) ;
INSERT INTO incasfact VALUES (1236, 1117, 9754) ;
INSERT INTO incasfact VALUES (1236, 1118, 100000) ;
INSERT INTO incasfact VALUES (1236, 1120, 7315) ;
INSERT INTO incasfact VALUES (1237, 1117, 9754) ;
```

```
INSERT INTO incasfact VALUES (1238, 1113, 106275) ;
INSERT INTO incasfact VALUES (1239, 1117, 3696) ;
```

3.5. Tranzacții

În câteva dintre scripturile de populare a tabelelor, prezentate în paragrafele anterioare, a fost strecurată, pe final, o comandă discretă – COMMIT. Şi în limba română, mai ales în jargonul orăşenesc, a apărut termenul "a comis-o !", în sensul că a gafat-o fără dubii/întoarcere. Oricât ar parea de ciudat, termenul se învecinează cu mai sus amintita comandă.

3.5.1. Principiile de bază ale tranzacțiilor

După cum afirmam în capitolul 1, tranzacțiile grupează comenzi care operează pe principiul *toți pentru unul, unul pentru toți*. Comenzile incluse în tranzacții sunt de tip DML (INSERT, UPDATE sau DELETE). Începutul tranzacției poate fi implicit (în Oracle, prima comandă DML dintr-o sesiune de lucru marchează, automat, începutul unei tranzacții; la fel prima comandă care urmează sfârşitului unei tranzacții) sau explicit, prin comanda BEGIN TRANSACTION sau START TRANSACTION. Încheierea cu succes a unei tranzacții este marcată prin comanda COMMIT[37]. Încheierea eronată şi, implicit, anularea efectului tuturor comenzilor de la începutul tranzacției până în momentul curent se realizează prin ROLLBACK.

Să luăm cazul cel mai des invocat atunci când se încearcă a se explica principiile de bază ale unei tranzacții: plata, de la un bancomat, a facturii telefonice. Plătitorul are un cont – ContPlătitor la banca X. Plata trebuie făcută în contul operatorului de telefonie – ContBeneficiar deschis la banca Y (X poate fi egal cu Y, dar asta nu are importanță pentru discuția noastră). Iată scenariul:

1. Autentificarea plătitorului (introducere card, apoi tastare PIN);
2. Alegerea, folosind meniul bancomatului, a operațiunii, apoi tastarea sumei şi contului client; pe baza acestor elemente, aplicația identifică ContPlătitor şi ContBeneficiar;

[37] Unele servere BD (cum este Oracle) încheie (cu succes) o tranzacție, implicit, la execuția unei comenzi DDL.

3. Se adaugă în baza de date în care se află ContPlătitor înregistrări care consemnează momentul plății, bancomatul de la care s-a făcut plata etc.
4. Se scade Suma din ContPlătitor;
5. Se adaugă în baza de date în care se află ContBeneficiar informații privind plata (data/ora încasării, modul de încasare, sursa banilor – ContPlătitor) etc.;
6. Se adună Suma la ContBeneficiar;
7. Se eliberează chitanța.

Tranzacția propriu-zisă începe cu pasul 3 şi se finalizează cu pasul 6. Trecerea la pasul următor este condiționată de confirmarea pasului curent. Dacă oricare din paşii 3-6 generează o eroare (se pierde legătura cu serverul pe care este stocat ContPlătitor, sau suma disponibilă în ContPlătitor este mai mică decât suma de plătit etc.), atunci întreaga tranzacție este abandonată. Dacă, însă, şi pasul 6 a decurs cu succes, tranzacția este încheiată cu succes.

Cea mai tristă ar fi situația în care Suma este scăzută din ContPlătitor, dar nu poate fi adăugată în ContBeneficiar (a picat rețeaua de comunicații!). Plătitorul ar avea conştiința curată (din punctul de vedere al plății), în timp ce pentru operatorul de telefonie ar fi un rău platnic. Ei bine, gruparea paşilor în tranzacții ar trebui să prevină asemenea gen de probleme.

Pentru a vă face o idee asupra importanței tranzacțiilor, trebuie să avem în vedere că toți aceşti paşi trebuie să se deruleze în condițiile în care, simultan, accesează şi modifică baza de date un număr foarte mare de utilizatori. Să presupunem că plătitorul are un acord cu banca, prin care plata facturii la gaz metan se face automat la o anumită dată într-un anumit cont – ContBeneficiar2. Este posibil ca, tocmai în momentele în care posesorul cardului meştereşte la bancomat încercând să-şi plătească factura la telefon, aplicația de plată automată a facturii la gaz să debiteze ContPlătitor, după un scenariu similar celui de mai sus. Ei bine, dacă prima tranzacție (cea cu plata facturii telefonice) este începută, a doua tranzacție nu "vede" nicio modificare în soldul ContPlătitor până în momentul finalizării cu succes a primeia[38].

În standardele SQL, comenzile legate de controlul tranzacțiilor sunt:

[38] Problema gestiunea tranzacțiilor şi a simultaneității (concurenței) este una la mare preț în manualele de baze de date. Pentru o tratare in extenso, inclusiv cele patru proprietăți (ACID) şi nivelurile de izolare, vezi [Dollinger 1998], pp.183-254, [Ionescu2004], pp.237-280

- START TRANSACTION – permite pornirea şi, eventual, configurarea unei tranzacţii;
- SET TRANSACTION – stabileşte parametrii următoarei tranzacţii pentru sesiunea curentă;
- SET CONSTRAINTS – stabileşte regimul restricţiilor pentru tranzacţia curentă (dacă este începută) sau pentru următoarea/următoarele (când nu există o tranzacţie curentă);
- SAVEPOINT – declară un punct de salvare; într-o tranzacţie pot fi mai multe puncte de salvare, pentru ca, în caz de erori, să nu piardă efectul tuturor comenzilor din tranzacţie, ci până la un anumit SAVEPOINT;
- RELEASE SAVEPOINT – anulează un punct de salvare;
- COMMIT – încheie (cu succes) tranzacţia curentă;
- ROLLBACK – încheie (cu eşec) tranzacţia curentă, refăcând starea BD dinaintea începerii tranzacţiei curente.

Dintre cele patru servere, DB2 şi Oracle prezintă tranzacţii implicite, în timp în PostgreSQL tranzacţiile trebuie declarate prin comanda BEGIN sau prin comanda START TRANSACTION. SQL Server permite şi alocarea unui nume pentru fircare tranzacţie. Tranzacţiile explicite încep cu BEGIN TRANSACTION şi se încheie fie prin COMMIT TRANSACTION sau ROLLBACK TRANSACTION.

3.5.2. Restricţii amânabile

Restricţiile declarate într-o tabelă sunt verificate imediat dupa o comandă INSERT sau UPDATE. Lucrurile pot fi, însă, uşor modificate. Să revenim la cazul facturilor, cu mici schimbări. Listingul 3.15 crează două tabele, EX_FACT şi EX_FACT_DETALII, care sunt versiuni *light* ale tabelelor FACTURI şi LINIIFACT. Sintaxa este cea a dialectului SQL din Oracle.

Listing 3.15. Script Oracle de testare a unor restricţii neamânabile

```
DROP TABLE ex_fact_detalii ;
DROP TABLE ex_fact ;

CREATE TABLE ex_fact (
    NrFact NUMBER(6) NOT NULL PRIMARY KEY,
    DataFact DATE DEFAULT CURRENT_DATE NOT NULL,
    NrLinii NUMBER(2) DEFAULT 0 NOT NULL
        CONSTRAINT ck_linii CHECK (NrLinii > 0 )
    ) ;

CREATE TABLE ex_fact_detalii (
    NrFact NUMBER(6) NOT NULL REFERENCES ex_fact (NrFact),
    Linie NUMBER(2) NOT NULL CONSTRAINT ck_linie_fd CHECK (Linie > 0),
    Produs VARCHAR2(40) NOT NULL,
    Cantitate NUMBER(8) NOT NULL,
    PretUnit NUMBER (9,2) NOT NULL,
    CONSTRAINT pk_fact_detalii PRIMARY KEY (NrFact,Linie),
    CONSTRAINT un_fact_detalii UNIQUE (NrFact, Produs)
    ) ;
```

În tabela EX_FACT există, printre altele, un atribut calculat care "spune" câte linii are factura respectivă[39]. Interesul nostru este ca să nu existe facturi fără nicio linie (în FACTURI există asemenea situații nedorite), astfel încât am definit restricția CK_LINII potrivit căreia valoarea atributului NrLinii trebuie să fie strict mai mare ca zero. Cum e şi normal, atunci când inserăm prima înregistrare în tabela EX_FACT: *INSERT INTO ex_fact (NrFact) VALUES (123456)*, ne pricopsim cu un mesaj de eroare – vezi figura 3.9 – întrucât valoarea implicită a NrLinii este 0.

Modificăm tipul restricției CK_LINII, adăugând în comanda de creare – vezi listing 3.16 - două clauze, DEFERRABLE şi INITIALLY DEFERRED. Prima semnalizează că restricția cu pricina este *amânabilă*, adică poate fi amânată până la finalul tranzacției. Cealaltă clauză indică faptul că restricția este nu numai amânabilă, dar şi *amânată*. Aceasta deoarece, ulterior, dacă vrem să trecem o restricție pe starea "clasică" (verificare imediată, adică chiar în momentul INSERT-ului sau UPDATE-ului), folosim clauza *SET CONSTRAINT ck_linii IMMEDIATE*.

Vă întrebați, probabil, de ce a fost nevoie să ştergem şi apoi să recreăm tabelele. Ei bine, am fost forțați. În Oracle, o restricție neamânabilă nu poate fi modificată în amânabilă.

[39] Cotrolul valorii acestui atribut este posibil prin declanşatoare – vezi capitolul 17.

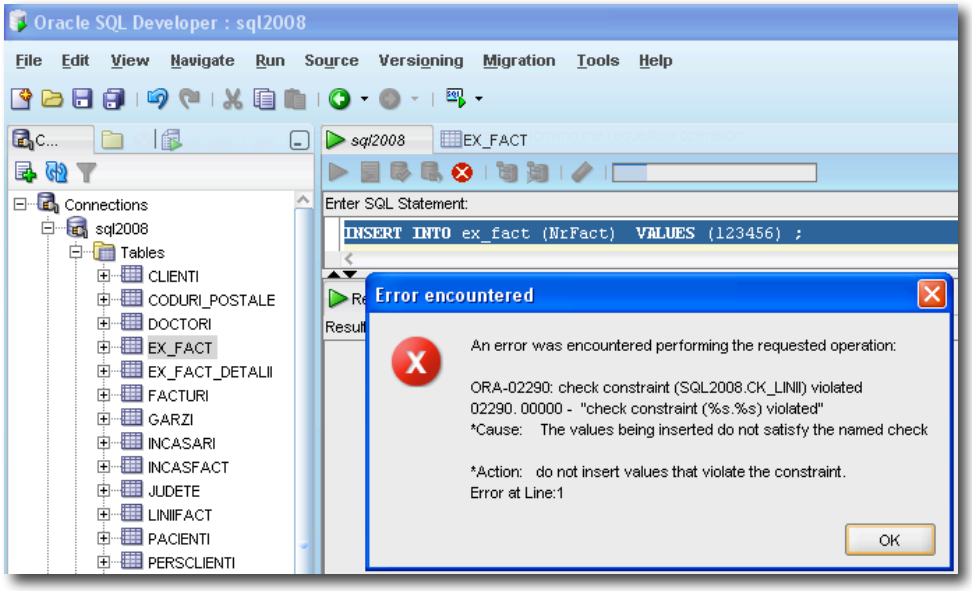


Figura 3.9. Violarea restricţiei (ne-amânabile) CK_LINII

Listing 3.16. Script Oracle de testare a unei restricţii amânabile/amânate

```
DROP TABLE ex_fact_detalii ;
DROP TABLE ex_fact ;

CREATE TABLE ex_fact (
    NrFact NUMBER(6) NOT NULL PRIMARY KEY,
    DataFact DATE DEFAULT CURRENT_DATE NOT NULL,
    NrLinii NUMBER(2) DEFAULT 0 NOT NULL
        CONSTRAINT ck_linii CHECK (NrLinii > 0 ) DEFERRABLE INITIALLY DEFERRED
    ) ;

CREATE TABLE ex_fact_detalii (
    NrFact NUMBER(6) NOT NULL REFERENCES ex_fact (NrFact),
    Linie NUMBER(2) NOT NULL CONSTRAINT ck_linie_fd CHECK (Linie > 0),
    Produs VARCHAR2(40) NOT NULL,
    Cantitate NUMBER(8) NOT NULL,
    PretUnit NUMBER (9,2) NOT NULL,
    CONSTRAINT pk_fact_detalii PRIMARY KEY (NrFact,Linie),
    CONSTRAINT un_fact_detalii UNIQUE (NrFact, Produs)
    ) ;

INSERT INTO ex_fact (NrFact)  VALUES (123456) ;

INSERT INTO ex_fact_detalii VALUES (123456, 1, 'Un produs', 100, 50) ;

UPDATE ex_fact SET NrLinii = NrLinii + 1 WHERE NrFact = 123456 ;

COMMIT ;
```

Comenzile INSERT şi UPDATE din script funcționează fără probleme. Dacă, însă, uităm comanda UPDATE care incrementează NrLinii, la execuția comenzii COMMIT se generează eroarea din figura 3.10.

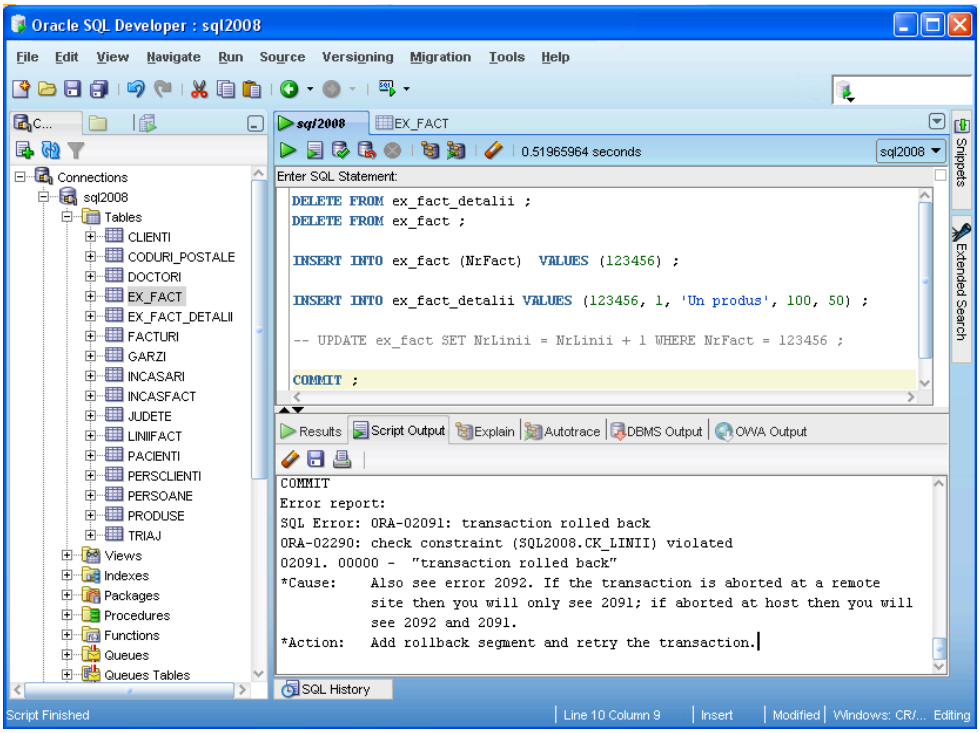


Figura 3.10. Violarea restricţiei (amânabile) CK_LINII

Nu întâmplător am exemplificat restricțiile amânabile în Oracle. Nici DB2 şi nici SQL Server nu au implementat mecanismul, iar în PostgreSQL doar restricțiile referențiale pot fi amânate.

3.5.3. Tabele temporare globale şi locale

O tabelă temporară, globală sau locală are o durată de viață limitată fie la o sesiune de lucru, fie la o tranzacție. Detalii despre comenzile de creare şi opțiunile de lucru cu tabelele temporare sunt prezentate în capitolul 14.