Proiect SGBD

–

Gestiunea Unui lanţ de Magazine

Theodor Moroianu

December 30, 2020

Contents

1 Introducere

Pentru proiectul meu final în cadrul cursului de Sisteme de Gestiune a Bazelor de Date am decis să ies puţin din tiparul obisnuit, şi să fac o baza de date mai mare şi umpluta cu date verosimile. In acest scurt referat o să prezint paşii pe care i-am urmat pentru a îmi construi baza de date, ideea din spatele ei, si cateva exemple de potenţială utilizare.

2 Ideea din Spatele Proiectului

M-am gandit la mai multe idei de proiect (mai multe scenarii / situatii pe care sa le modelez cu o baza de date.

Printre ele, cele care mi-au placut cel mai mult au fost:

- Gestiunea unei universitati. Aceasta cuprinde:
  - Lista cu profesorii, studentii, cursurile si salile din universitate.
  - Tabele asociative intre profesori si cursuri, studenti si cursuri, si cursuri si salile din universitate.
  - Istoric al universitatii – ce fosti studenti are universitatea, ce cursuri a luat fiecare student, ce note a avut etc.
  - Orice alte informatii care depind de universitate.
- Gestiunea unei primarii. Gestiunea primariei ar cuprinde:
  - Gestiunea functionarilor – ce job au, unde lucreaza, cine e responsabil pentru ei etc.
  - Gestiunea budgetului – O lista de potentiale achizitii, cu importanta si prioritatea lor.
  - Lista cu strazile, parcurile si casele din sector.
  - Toate autorizatiile de constructie.
  - Orice alte elemente pe care le gestioneaza primaria.
- Gestiunea unui lant de magazine. Trebuie tinute:
  - Lista de magazine (cu locatia, manager-ul, produsele etc).
  - Lista de angajati.
  - Lista de clienti si de furnizori.
  - Istoric al cumpararilor.

Dupa mai multe zile de gandire, am decis sa merg pe a 3-a varianta.
Ideea proiectului este:

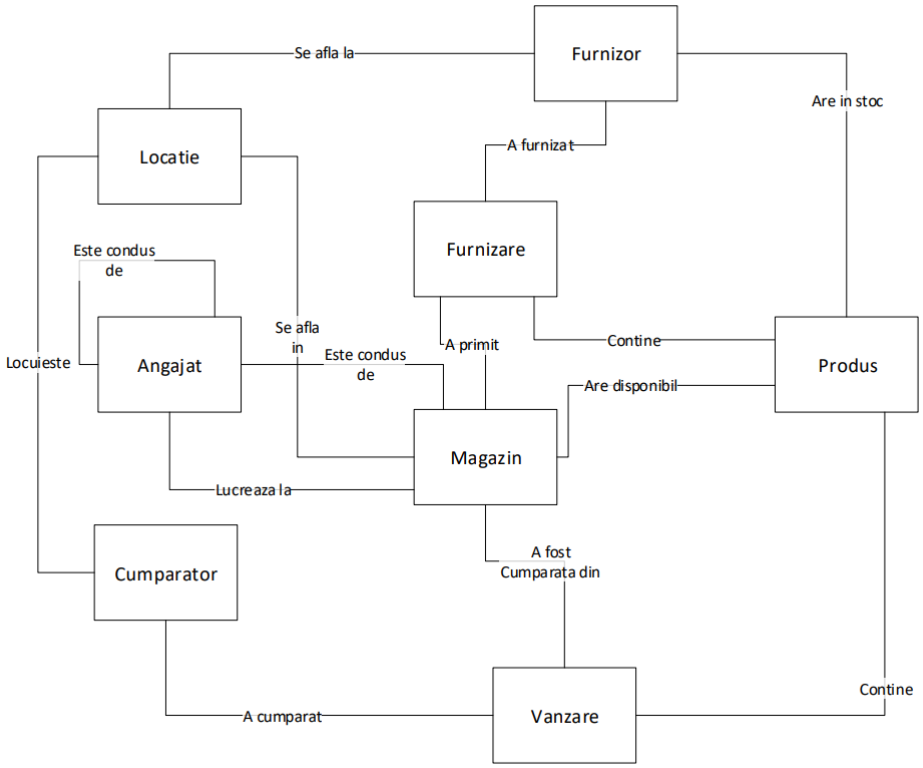
1. De-a prezenta un potential model cat mai complet pentru functionarea unui lant de magazine.
2. De-a modela diagrama *E/R* si schema relationala, cu lista completa de coloane, pentru a usura intelegerea bazei de date.
3. De-a implementa baza de date in *SQL*.

4. De-a arata cateva exemple de utilizare cu cod de *PL/SQL* care sa execute diferite operatii pe baza de date.

5. de-a pune la dispozitia utilizatorului un script capabil sa ne populeze tabelele cu informatii aleatoare dar plauzibile.

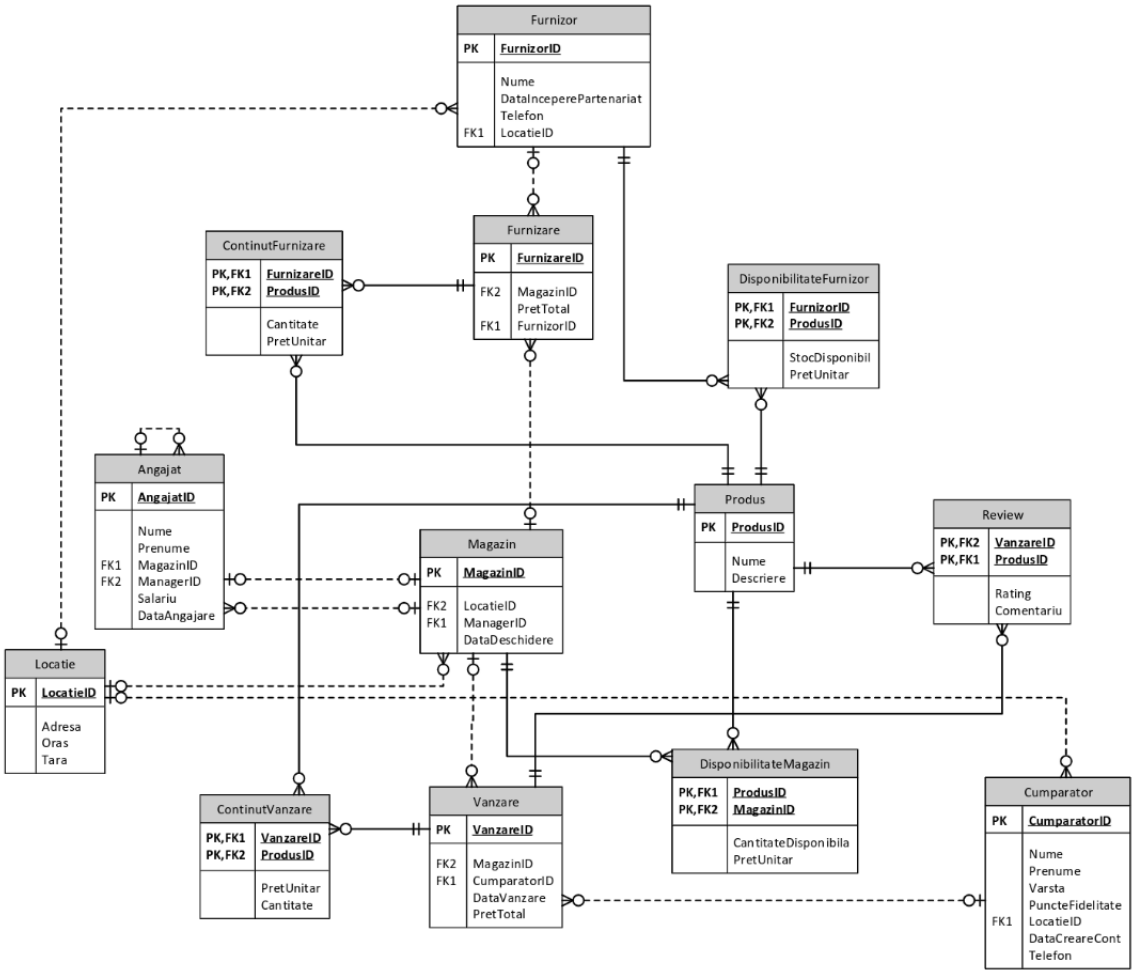
3 Diagrama Entitate-Relatie

Diagrama Entitate-Relatie este data de modelul de mai jos:



4 Schema Relatională

Schema relatională este data de modelul de mai jos. Modelul este identic cu cel din cazul diagramei entitate relatie, exceptand tabelele asociative create pentru a putea rezolva relatile *many-to-many*.



5 Codul SQL de Creare a Bazei de Date

Codul de creare a bazei de date este in mare parte oarecum trivial. De exemplu, pentru a crea tabelea *Vanzare*, avem urmatoarea cerere *SQL*:

```
CREATE TABLE vanzare (
    vanzareID NUMBER PRIMARY KEY,
    magazinID NUMBER,
    cumparatorID NUMBER,
    dataVanzare DATE,
    pretTotal NUMBER NOT NULL,
    FOREIGN KEY (magazinID) REFERENCES
        magazin(magazinID) ON DELETE SET NULL,
```

```
    FOREIGN KEY (cumparatorID) REFERENCES
        cumparator(cumparatorID) ON DELETE CASCADE);
```

In schim, uitandu-ne mai atenti la cheile externe dintre tabelul *Angajat* si *Magazin*, obervam ca ambii, pentru a isi satisface constrangerile de integritate, au nevoie de existenta celuilalt tabel.

Pentru a rezolva acesta situatie de deadlock, declaram un tabel fara constrangerea de cheie externa, pe care o adaugam ulterior.

Astfel, avem urmatorii pasi:

- Cream tabloul *Angajat*, fara cheia externa spre *Magazin*:

```
CREATE TABLE angajat (
    angajatID NUMBER PRIMARY KEY,
    nume VARCHAR2(50),
    prenume VARCHAR2(50),
    magazinID NUMBER,
    managerID NUMBER,
    salariu NUMBER,
    dataAngajare Date,
    FOREIGN KEY (managerID) REFERENCES
        angajat(angajatID) ON DELETE SET NULL);
```

- Cream tabloul *Magazin*:

```
CREATE TABLE magazin (
    magazinID NUMBER PRIMARY KEY,
    locatieID NUMBER,
    managerID Number,
    dataDeschidere Date,
    FOREIGN KEY (managerID) REFERENCES
        angajat(angajatID) ON DELETE SET NULL,
    FOREIGN KEY (locatieID) REFERENCES
        locatie(locatieID) ON DELETE SET NULL);
```

- Adaugam tabelului *Angajat* constrangerea de cheie externa:

```
ALTER TABLE angajat
    ADD CONSTRAINT angajatfkmagazin
        FOREIGN KEY (magazinID) REFERENCES
            magazin(magazinID);
```

6 Popularea Bazei de Date

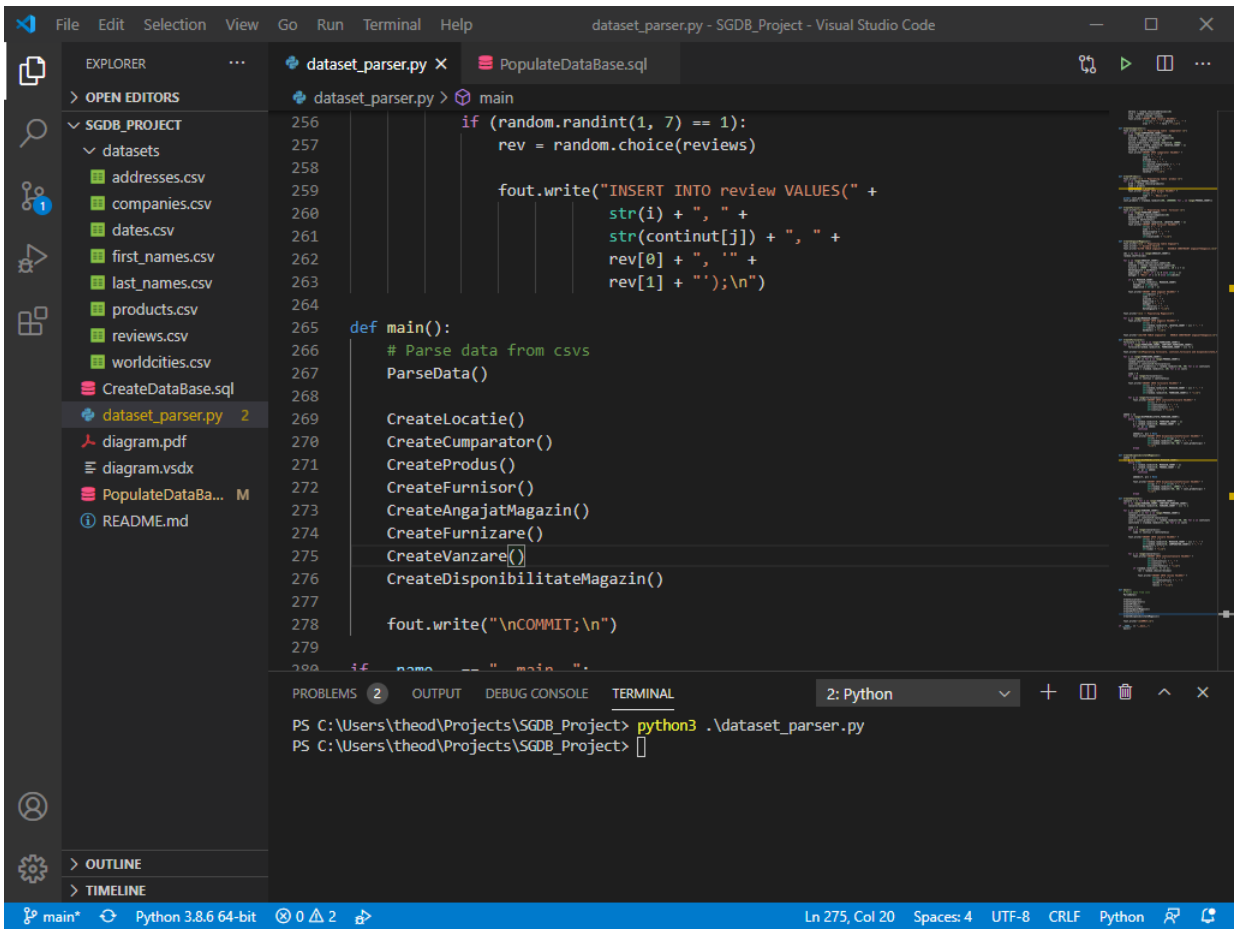
Am incercat sa caut date reale pentru modelul meu, dar evident nu am gasit nimic. Potentialele motive pentru care nu exista astfel de date *"In the wild"* sunt destul de evidente:

- Probleme legate de siguranta datelor (GDPR etc)
- Probleme legate de competite / secretizare
- Modelele reale sunt mult mai complexe decat al meu

Totusi, nu doream sa imi populez baza de date cu "3-5 inregistrari" asa cum cerea proiectul. Asadar, am stat cateva ore pe Kaggle.com, un website cu dataseturi din aproape oricare domeniu, si mi-am salvat:

- Un dataset cu lista celor mai comune prenume din SUA
- Un dataset cu lista celor mai comune nume de familie din SUA
- Un DS cu o lista de adrese, orase si tari
- Un DS cu o lista de firme (nu doar producatori, dar macar sunt nume coerente de firme)
- O lista cu produse vandute online
- O lista cu reviewuri.

Folosind aceste date, mi-am facut un script in *Python*, cu care mi-am generat un script de *SQL* care populeaza baza de date cu un numar arbitrar de intregistrari, generate aleator. Pentru a nu incetini prea mult baza de date, am decis sa adaug numai 12.000 de integistrari, desi puteam doar modificand cateva constante sa adaug milioane.



Datele sunt inserate in baza de date una cate una, care probabil ca nu este cel mai eficient mod, dar este cel mai usor de implementat.

7 Cerinţa VI

Pentru cerinta 6 am decis sa rezolv o problema care apare in urmatorul scenariu:

Compania vrea sa organizeze o campanie de promotie de craciun. Pentru aceasta campanie, compania doreste sa ii trimita fiecarui cumparator un mesaj de tipul "De cand nu ai mai fost pe la noi produsele pe care le-ai cumparat ultima data s-au ieftinit: Ai cumparat produsul X la pretul Y dar acum il poti cumpara la pretul Z ...".

Evident, campania aceasta este facuta doar pentru a atrage clientii, deci presupunem ca pretul unui produs este costul cel mai ieftin al produsului in oricare dintre magazinele in care este in stoc.

Conform cerintelor, subprogramul definit in fisierul "Cerinta_Nr_6.sql" foloseste un tip de colectii studiat.

8 Cerinţa VII

Cerinta 7 rezolva urmatorul scenariu:

Compania a decis sa dea bonus de craciun angajatilor.

Evident, nu are asa multi bani de cheltuit, asa ca doreste sa cheltuie cat mai putini. Astfel compania a decis sa premieze un angajat, subordonatii sai directi, subordonatii acestora etc. In termeni tehnici compania doreste sa premieze un subarbore din arborele angajatilor.

Premiul consta din cresterea salariului cu X%, unde X este ales de CEO (pe ascuns, ca sa nu comenteze lumea ca e prea mic). Pe de alta parte, ca sa nu comenteze angajatii, trebuie sa fie premiati cel putin Y angajati. Care sunt cele mai bune Z alegeri de premiere a angajatilor?

Conform cerintelor, subprogramul definit in fisierul "Cerinta_Nr_7.sql" foloseste un tip de cursoare studiat.

9 Cerinţa VIII

Cerinta 8 rezolva urmatorul scenariu:

Compania a decis sa isi verifice stocul diferitor produse. Pentru acesta, a rugat departamentul de IT sa puna managerilor magazinelor la dispozitie o functie de SQL, care sa efectueze urmatoarele calcule:

- Managerul isi introduce ID-ul, magazinul pe care il conduce si ID-ul produsului de care este interesat.
- Functia se asigura ca managerul este intr-adevar manager in magazinul cu ID-ul mentionat (cum functia de SQL este folosita in cadrul altor aplicatii putem sa presupunem ca un angajat nu poate introduce alt ID decat al sau). –
- Daca managerul este validat, atunci functia intoarce cantitatea disponibila a produsului respectiv.

Conform cerintelor, subprogramul definit in fisierul "Cerinta_Nr_8.sql" foloseste o functie stocata, care foloseste 3 tabele din baza de date.

10 Cerinţa IX

Cerinta 9 rezolva urmatoatorul scenariu:

Compania a decis sa faca o noua campanie promotionala.

Astfel, pentru fiecare cumparator trebuie sa afisam urmatorul mesaj:

"Draga XXXX, pe data de YYYY ai cumparat produsul ZZZZ la pretul VVV – care are un review mediu de TTTT, si il poti cumpara la un pret de UUUU".

Bine inteles, pentru a face un astfel de mesaj trebuie ca produsul sa aiba cel putin un review, si sa fie disponibil in cel putin un magazin.

11 Cerinţa X

Cerinta 10 rezolva urmatoatorul scenariu:

Pentru a asigura putina integritate in baza de date, compania doreste ca dupa orice modificare a structurii de angajat / şef, sa se verifice daca toti angajatii raman in continuare subordonati directi sau indirecti ai sefului.

Altfel spus, dupa fiecare modificare a tabelului *Angajat* trebuie verificat ca relatiile de subordonare au o structura arborescenta, un singur angajat fiind "seful" tututor.

12 Cerinţa XI

Cerinta 11 rezolva urmatoatorul scenariu:

Pentru a nu avea conflicte interne, directorul doreste ca la modificarea salariului unui angajat, acesta sa nu se modifice cu mai mult de 10%. Implementam un trigger care verifica fiecare modificare a tabelului angajat, la nivel de linie.

13 Cerinţa XII

Cerinta 12 rezolva urmatoatorul scenariu:

Pentru a facilita gasirea problemelor in baza de date, se doreste crearea unui trigger, care sa salveze informatii despre eventuale modificari ale bazei de date.

De asemenea, pentru a evita greseli datorate oboselii, se doreste ca adaugarea / stergerea / modificarea tabelelor sa nu fie posibila inafara programului de lucru (8:00 - 17:00 de luni pana vineri).

14 Resurse Utilizate

Pe parcursul acestui proiect am folosit / creat mai multe resurse. Lista resurselor si documentelor folosite este:

- Referat pentru curs – *"Populatea Randomizata A Unei Baze De Date"*
- Referat pentru curs – *"Algoritmi Procedurali Intr-o Lume Declarativa"*

- Repo de *Github* in care am lucrat la proiect
- Kaggle.com – Un website cu dataseturi gratuite