CZU: 004.43

APLICAREA ORM LA REZOLVĂRI DE PROBLEME DIN ECONOMIE

Vitalie ȚÎCĂU, lector universitar
https://orcid.org/0000-0002-2032-3662
Universitatea de Stat „Alecu Russo” din Bălți

Rezumat. Logica Object Relational Mapping este capacitatea de a scrie interogări și de a manipula date folosind o paradigmă orientată pe obiecte. O bază de date relațională conține rânduri care sunt stocate în tabele. Cu toate acestea, atunci când se scrie un program într-un limbaj de nivel înalt orientat pe obiecte, este mult mai convenabil când datele preluate din baza de date pot fi accesate sub formă de obiecte. Tehnologia ORM ajută la operarea rapidă cu informațiile din baza de date, o metodă performantă de lucru cu cererile și tabelele. În lucrare este cercetată abordarea utilizării tehnologiei ORM la elaborarea unei aplicații de gestiune a datelor unei companii.
Cuvinte-cheie: ORM, baze de date, POO, Python, aplicații în practică.

APPLYING ORM TO PROBLEM SOLVING IN ECONOMICS

Abstract. Object Relational Mapping logic is the ability to write queries and manipulate data using an object-oriented paradigm. A relational database contains rows that are stored in tables. However, when writing a program in a high-level object-oriented language, it is much more convenient when the data retrieved from the database can be accessed as objects. ORM technology helps to quickly operate with database information, a high-performance method of working with requests and tables. The paper researches the approach of using ORM technology to develop a company's data management application.
Keywords: ORM, databases, OOP, Python programming, applications in practice.

Introducere

Limbajele de programare orientate pe obiecte reprezintă datele într-un graf interconectat de obiecte, pe când bazele de date relaționale folosesc un mod tabelar de reprezentare. Efortul de a conecta atributele claselor definite prin intermediul unui limbaj orientat pe obiecte cu câmpurile tabelelor din baza de date nu poate fi ignorat, iar scopul unui ORM este acela de a crea o relație naturală, transparentă, fiabilă și de durată între cele două modele. Această nepotrivire de paradigmă pare să nu își fi găsit încă o soluționare definitivă care să fie aprobată de toți actorii din industria IT, însă opinia generală este aceea că framework-urile ORM reprezintă un important pas înainte.

Fiind un limbaj de programare bine conceput, Python este perfect pentru rezolvarea problemelor din lumea reală, cum ar fi cele pe care dezvoltatorii trebuie să le rezolve zilnic. Este folosit în cea mai largă gamă de aplicații – atât ca unealtă pentru gestionarea altor componente software, cât și pentru implementarea programelor independente.

În lucrare sunt cercetate modurile de aplicare a limbajului Python la elaborarea de aplicații și componentele sale distinctive, sunt analizate etapele de proiectare a unei baze de date, pornind de la planificarea bazei de date, alegerea unui sistem de gestiune, cât și implementarea acesteia in cadrul sistemelor respective. În lucrare se descriu pas cu pas

conceptele teoretice și ideile practice pentru a putea crea și dezvolta o bază de date corectă și funcțională. De asemenea este descris modul de elaborare a aplicației de gestiune a datelor unei companii utilizând conceptele ORM și baze de date PostgreSQL.

Metode și materiale aplicate

ORM este o tehnologie de programare care conectează baze de date cu concepte de limbaje de programare orientate pe obiecte, creând o „bază de date de obiecte virtuale". Există atât implementări proprii cât și gratuite ale acestei tehnologii [1].

ORM face posibilă accesarea și manipularea obiectelor fără ca programatorii să fie interesați de sursa de date de unde provin aceste obiecte. Această tehnică a apărut din necesitatea de a depăși diferențele de paradigmă dintre modelul orientat pe obiecte (susținut de limbajele de programare de nivel înalt actuale) și modelul relațional (utilizat de cele mai populare sisteme de gestiune a bazelor de date).

Procesul automat de stocare a obiectelor într-o bază de date relațională folosind un framework ORM, constă în maparea obiectelor la tabelele corespunzătoare, asocierea dintre ele fiind descrisa folosind metadata. Un framework ORM complet include următoarele funcționalități:

- un API pentru operațiile CRUD (create, read, update, delete) aferente claselor persistente;
- un limbaj pentru specificarea interogărilor adresând clasele persistente și atributele acestora;
- un mod care să faciliteze definirea metadata pentru mapările dintre obiect și tabel;
- o abordare consistentă a tranzacțiilor, a metodelor de stocare a datelor și a asocierilor dintre clase;
- tehnici de optimizare în funcție de natura aplicației.

Avantajele utilizării ORM-urilor sunt:

- economia de timp pentru utilizatorii non-SQL – se poate accelera procesul cu limbajul Python;
- ORM face comutarea între diferite baze de date relaționale (de exemplu, de la MySQL la PostgreSQL) mult mai ușoară, deoarece majoritatea funcțiilor rămân aceleași;
- unele dintre interogări au rezultate mai eficiente, decât dacă se scriu individual.

Dezavantajele utilizării ORM-urilor sunt:

- deoarece ORM este un concept diferit, pentru a utiliza un ORM, este necesar timp suplimentar pentru a-l studia;
- pentru practicienii SQL, utilizarea ORM poate fi mai puțin eficientă și utilă, deoarece nu oferă atât de multe oportunități comparativ cu scrierea individuală a interogărilor;
- conflictele suplimentare pot apărea la setarea configurației etc.

ORM-urile Python des utilizate sunt [2]: Django ORM, SQLAlchemy ORM, Peewee ORM, Pony ORM, SQLObject etc. Caracteristicile de bază ale *Django* sunt:

- SEO optimizat;
- este extrem de rapid;
- dispune de un cadru complet încărcat care vine împreună cu autentificări, administrări de conținut, fluxuri RSS etc.;
- este foarte sigur ajutând astfel dezvoltatorii să evite greșelile obișnuite de securitate, cum ar fi falsificarea cererilor între site-uri (csrf), clickjacking, scripturi între site-uri etc.;
- este extrem de scalabil, ceea ce, la rândul său, ajută la satisfacerea celor mai grele cereri de trafic;
- este imens de versatil, care permite de a dezvolta orice fel de site-uri web.

Django urmează arhitectura MVT (*Model View Template*), care se bazează pe arhitectura MVC (*Model View Controller*). Principala diferență dintre aceste două este că Django în sine se ocupă de partea controlerului.

SQLAlchemy este un Python ORM bine apreciat, deoarece obține nivelul de abstractizare „corect” și pare să facă interogări ale bazelor de date complexe mai ușor de scris decât Django ORM în majoritatea cazurilor.

Peewee ORM este o implementare ORM Python care este scrisă pentru a fi „mai simplă, mai mică și mai accesibilă” decât SQLAlchemy.

Pony ORM este un alt Python ORM disponibil ca sursă deschisă, sub licența Apache 2.0 și permite dezvoltatorilor să lucreze cu conținutul unei baze de date sub formă de obiecte. Pony ORM este o bibliotecă pentru limbajul Python care permite de a lucra convenabil cu obiecte care sunt stocate ca rânduri într-o bază de date relațională.

Pony are câteva avantaje distincte comparativ cu alte ORM, cum ar fi Django și SQLAlchemy:

- sintaxă excepțional de convenabilă pentru scrierea interogărilor;
- optimizare automată a interogărilor;
- soluție elegantă pentru problema *N+1;*
- editorul de schemă a bazei de date online.

Limbajul de programare Python poate fi folosit pentru a rezolva o gamă mai largă de probleme.

Proiectarea unei baze de date reprezintă un proces ce implică dezvoltarea și rafinarea structurii unei baze de date pe baza cerințelor formulate de către beneficiarul bazei de date și a cerințelor deduse pe baza efectuării analizei domeniului ce urmează a fi modelat. Principalul obiectiv urmărit la proiectarea unei baze de date este asigurarea consistenței, integrității și preciziei datelor.

Dacă proiectul unei baze de date este incorect, atunci este dificil de extras informațiile necesare, fiind posibilă obținerea de informații false [3].

Planificarea bazei de date reprezintă controlul activităților ce permit realizarea efectivă și eficientă a etapelor de proiectare a unei baze de date. În acest scop se realizează următoarea documentație:

- identificarea scopului bazei de date;
- obiectivele bazei de date.

Proiectarea bazei de date cuprinde următoarele trei etape:

- proiectarea conceptuală a bazei de date; - proiectarea logică a bazei de date;
- proiectarea fizică a bazei de date.
- proiectarea conceptuală.

Fie, că în cadrul departamentului economic al companiei funcționează serviciul contabilitate care pe lângă celelalte atribuții de organizare și conducere a contabilității companiei, desfășoară și activitatea de evidență personal și calculul salariului. Din operațiunile de bază ale acestei activități cele mai reprezentative sunt:

- întocmirea de situații și evidențe de personal pe diverse categorii;
- întocmirea situațiilor și propunerilor pentru trimiterea personalului la cursuri de specializare și perfecționare;
- întocmirea documentelor premergătoare pensionării personalului;
- se fac studii cu privire la gradul de profesionalizare al personalului societății;
- se fac studii și se întocmesc propuneri pentru plata indemnizațiilor pentru trepte de vechime, pentru completarea studiilor ale personalului;
- se ocupă de întocmirea la timp și în volum complet a documentelor legale de muncă ale personalului societății;
- se înaintează către conducerea societății, propuneri de promovare a personalului pe funcții corespunzătoare cu pregătirea și vechimea acestora;
- se întocmesc situații și evidențe cu privire la pontajul angajaților și înaintarea acestora la oficiul de calcul în vederea stabilirii retribuției lunare a personalului;
- întocmirea de documente de acces în cadrul societății și legitimații specifice.

Analiza elementelor necesare aplicării bazei de date. Întrucât compania are perspectivă, ceea ce presupune mărirea numărului de angajați permanenți și colaboratori, a devenit necesară utilizarea unui sistem informatic pentru evidența personalului și calculul salariilor [4].

Pentru a realiza o bază de date a personalului companiei trebuie cunoscute și stocate următoarele date necesare: numele și prenumele, ID, codul numeric personal, nr. contract de muncă, data contract de muncă, data începerii activității, data încetării activității, motivul încetării activității, tipul de contract de muncă, norma zilnica ore/ zi, adresa, persoane aflate

în întreținere, studii, funcția, sexul, salariul de încadrare, deducerea de bază, deducerea pentru persoanele aflate în întreținere.

Pentru calculul salariilor, după analiza și studiul legislației în vigoare s-a hotărât stocarea următoarelor date: numele și prenumele, codul numeric personal, ore plătite (ore efectiv lucrate), ore lucrătoare în luna respectivă, sporuri, rețineri/

Documentele de intrare pentru acest sistem sunt: fișa de pontaj; lista cu reținerile; contractul de muncă; grila privind deducerile personale.

Rapoartele care se doresc a se obține în urma prelucrării datelor în cadrul sistemului informațional: statul de plata lunar, fișe fiscală 1 și 2, situație de tip concedii medical, situație de tip concedii legale, structura personalului (studii, sex, etc.), listă de avans chenzinal, situația fluctuației de personal, centralizator privind contribuțiile companiei la bugetul asigurărilor sociale aferente salariilor, salariile, în cazul companiei se calculează lunar, plătindu-se într-o singură tranșă lunară (lichidare).

Datele sunt organizate în 5 tabele ale bazei de date. Modelul relațional este prezentat în figura 1.

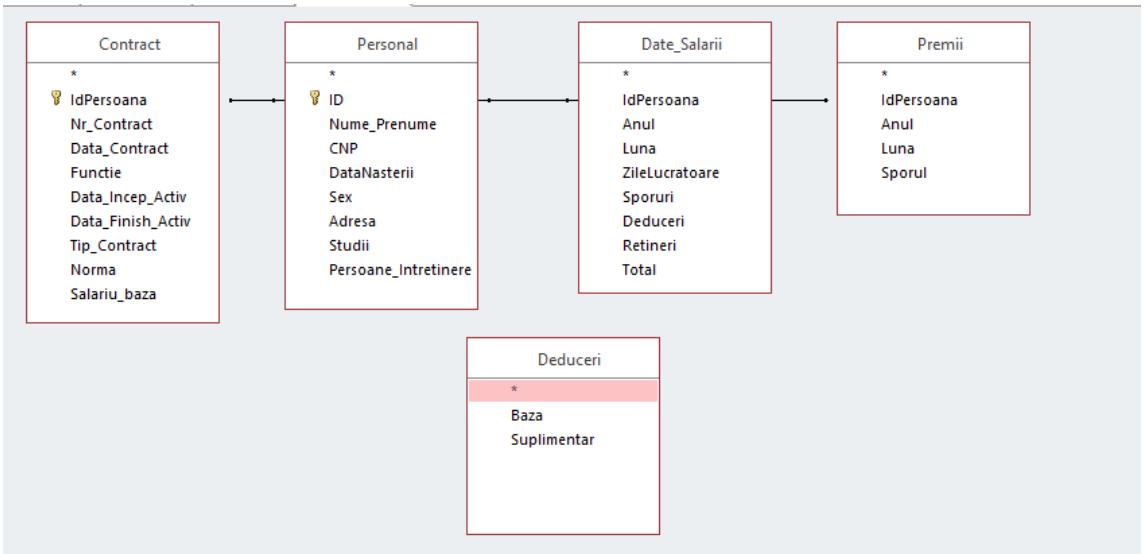

Figura 1. Modelul relațional al bazei de date

În aplicației sunt definite operațiile de: citire, adăugare, modificare, ștergere și căutare în tabelele bazei de date. De asemenea au fost definite funcții specifice pentru operațiile de:

- *actualizare* – crearea tabelelor bazei de date și înscrierea din fișiere text a datelor inițiale la moment;
- *afișare* – afișare datelor curente;
- *adăugare* – adăugarea de persoane, înscrierea contractelor sau atribuirea premiilor;
- *modificare* – modificarea / corectarea datelor eronate;
- *ștergere* – eliminarea articolelor din datelor eronate;

- *căutare* – căutarea unui angajat (unor angajați) după nume cu indicarea datelor contractuale:
- *calcul* – calculul salariului unui angajat într-o lună anumită și generarea unui raport de achitare a salariului într-un fișier text.

Concluzii

În lucrare sunt examinate și caracterizate principalele concepte de aplicare a ORM. Se poate spune, că nu există o soluție universală pentru strategia de data mapping.

Nu se poate afirma cu siguranță că un tip de ORM-uri este în toate aspectele mai bun decât altul. Se poate, însă, observa că unele ORM-uri se descurcă mai bine cu unele sarcini decât altele.

Nepotrivirea impedanței este un termen utilizat în combinație cu ORM și specifică dificultățile care apar la mutarea datelor între tabelele relaționale și obiectele aplicației. Modul în care un dezvoltator folosește obiecte este diferit de modul în care datele sunt stocate și unite în tabelele relaționale.

Pentru o utilizare fiabilă și corectă a unei baze de date este important să se urmărească pașii corecți în realizarea și proiectarea bazei. Important este ca fiecare etapă în elaborare să se parcurgă conform cerințelor acest lucru fiind într-un mod ierarhic.

Performanța ORM provine din traducerea codului aplicației într-o instrucțiune SQL corespunzătoare, care nu poate fi reglată tocmai corect, însă utilizarea unui framework ORM duce la scrierea mai rapida a codului, iar duplicările de cod sunt mai ușor de evitat.

Bibliografie

1. DALTON, A. *The Book of ORM*. 2015, 242 p. ISBN: 978-1909845770. [online]. Disponibil pe Internet [20.06.2021]. www.amazon.com/Book-Orm-J-Dalton/dp/1909845779
2. *Mapping Objects to Relational Databases: O/R Mapping In Detail*. [online]. Disponibil pe Internet [22.01.2022]. http://www.agiledata.org/essays/mappingObjects.html
3. RODIC, M. *Use SQLAlchemy with Django*. [online]. Disponibil pe Internet [25.01.2022]. https://rodic.fr/blog/sqlalchemy-django/
4. STANCIU, V.; GAVRILA, A.; MANGIUC D. *Sisteme informatice financiar-monetare,* București: Ed. ASE, 265 p.