UNIVERSITATEA TEHNICĂ GHEORGHE ASACHI IAȘI
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
SPECIALIZARE CALCULATOARE ȘI TEHNOLOGIA INFORMAȚIEI

DISCIPLINA BAZE DE DATE PROIECT

Catalog școlar online

Coordonator: Cătălin Mironeanu
Student: Scînteie Gabriel Alexandru
Grupa: 1307A
Iași, 2022

Titlul proiectului: Catalog școlar online

Analiza, proiectarea și implementarea unei baze de date care să stocheze și să modeleze informații despre clase, profesori, materii, elevi și notele acordate acestora, având ca scop ușurarea procesului educativ atât din perspectiva profesorilor cât și din perspectiva elevilor și a părinților acestora.

Descrierea cerințelor și model de organizare al proiectului

Baza de date propusă dorește digitalizarea atât a clasicului catalog fizic, cât și a informațiilor ce țin de profesori, clasă și sălile acestora. Acțiunile ce se permit sunt: adăugarea, modificarea și ștergerea informațiilor despre clase, elevi, profesori, materii și note. Astfel, profesorii pot adăuga note elevilor, iar copiii și părinții pot vedea oricând situația școlară. Pe lângă acest lucru, se pot afla diverse informații precum numele profesorilor, poziționarea claselor în spațiul școlii(etajul împreuna cu numărul sălii) și statistici precum media la o anumită materie sau media generală a unui elev.

Aplicația funcționează în baza următoarelor constrângeri:

- Fiecare clasă are o sală asociată iar toate orele corespunzătoare acelei clase se vor ține in acea sală
- Aplicatia nu isi propune stocarea unui orar pentru desfasurarea orelor
- Un profesor poate pune o nota unui elev doar dacă este profesor la clasa de care apartine elevul și doar la materiile pe care le predă el
- La o clasă aceeași materie poate fi predată de mai mulți profesori
- Adresele de mail trebuie să aibă un anumit format
- CNP-ul elevilor trebuie să aibă un anumit format
- Numele și prenumele elevilor și al profesorilor trebuie sa aibă macar două litere și să conțina doar litere
- Numărul unei clase trebuie să fie între 1 si 12, iar litera fiecărei clase este între A si E
- Școala are doar două etaje, pe fiecare etaj sunt câte 10 săli numerotate de la 1 la 10
- Nota este un număr întreg de la 1 la 10
- Denumirea materiei va avea cel puțin două litere și va conține doar litere
- Profesorii nu vor putea trece note în catalog pentru date calendaristice din viitor și nici pentru date calendaristice ce nu aparțin anului în curs

Informațiile de care avem nevoie sunt cele legate de:

- **Profesori:** numele, prenumele, materiile pe care le predau și clasele la care predau
- **Elevi**: codul matricol, numele, prenumele, CNP-ul și email-urile
- **Clase**: numele clasei, etajul și numărul sălii asignată clasei respective, numele elevilor ce aparțin clasei, numele materiilor ce se studiază la clasa respectivă și numele profesorilor ce predau la respectiva clasă
- **Note**: valoarea notei, data la care a fost pusă nota, codul matricol al elevului căruia îi este destinată nota, clasa din care face parte acesta, materia la care a fost notat elevul și profesorul ce l-a notat

Descrierea detaliată a entităților și a relațiilor dintre tabele

Entitățile din această aplicație sunt:

- Profesorii = entitate ce ține informațiile legate de profesori

  → ID_profesor(NUMERIC) = primary – key
  → Numele(VARCHAR(20)) = mandatory, constrangere lungime minim 2
  → Prenumele(VARCHAR(20)) = mandatory, constrangere lungime minim 2

- Materiile = entitate ce ține informațiile legate de materiile predate în școală

  → ID_materie(NUMERIC) = primary - key
  → Denumire(VARCHAR(30)) = mandatory, constrangere lungime minim 2

- Elevii = entitate ce ține informațiile despre elevi

  → CodMatricol(CHAR(4)) = primary - key
  → Nume(VARCHAR(20)) = mandatory, constrangere lungime minim 2
  → Prenume(VARCHAR(20)) = mandatory, constrangere lungime minim 2
  → CNP(CHAR(13)) = mandatory, unique key
  → EMAIL(CHAR(25)) = mandatory
  → CLASA(VARCHAR(3)) = relation UID, mandatory

- Înregistrările de note = entitate ce ține informațiile legate de punerea unei note

  → ID_inregistrareNota(NUMERIC) = primary – key
  → Nota(NUMERIC) = mandatory, constrangere valoare întreagă între 1 și 10
  → Data(DATE) = mandatory, constrangere data să aparțină anului în curs și să nu fie o dată din viitor

→ ID_Materie(NUMERIC) = primary – key, relation UID, mandatory
→ COD_MATRICOL(CHAR(4)) = relation UID, mandatory
→ ID_Profesor(NUMERIC) = relation UID, mandatory
→ Clasa(VARCHAR(3) = relation UID, mandatory

- Clasele = entitate ce ține informații legate de clase

→ Clasa(VARCHAR(3))= primary – key, constrângere clasele să fie în intervalul 1A – 12E

- Sălile = entitate ce ține informații legate de poziția claselor în cadrul școlii

→ Clasa(VARCHAR(3)) = primary – key, relation UID
→ Etaj(NUMERIC) = mandatory, constrangere să aibă una din valorile 0,1 sau 2
→ Numar(NUMERIC) = mandatory, constrangere să fie un număr de la 1 la 10
→ Există o constrângere unique pentru perechea (etaj, numar) pentru a nu exista două înregistrări ce se referă la aceeași clasă

- Profesor - Clasă = entitate ce reunește informațiile legate de profesori, materiile pe care le predau ei și clasele la care predau aceste materii)

→ ID_Materie(NUMERIC) = primary – key, relation UID
→ ID_Profesor(NUMERIC) = primary – key, relation UID
→ Clasa(VARCHAR(3)) = primary –key, relation UID
→ Datorită constrângerii primary-key compuse o înregistrare este în mod unic identificată de o clasa, un profesor și o materie

În proiectarea aplicației s-au identificat relații de 1:1, 1:n și n:1.

Între entitățile **Sală** și **Clasă** există o relație de 1:1, deoarece unei clase îi este asociată o singură sală unde vor veni profesorii pentru a-și ține orele la clasa respectivă. Nu este posibilă mutarea orei de curs în afara sălii asignate fiecărei clase. O sală este identificată în mod unic de către clasa ce își tine orele acolo.

Între entitățile **Clasă** și **Elev** există o relație de one-to-many, deoarece într-o clasă se află mai mulți elevi, însă un elev nu poate aparține în același timp mai multor clase.

Între entitățile **Elev** și **ÎnregistrareNota** este o relație one-to-many, deoarece un elev va primi mai multe note în cadrul mai multor materii, însă o notă îi este atribuită unui singur elev. O notă este unică și se identifică prin intermediul unui id.

Între entitățile **Profesor-Clasă** și **ÎnregistrareNotă** este o relație de one-to-many deoarece un profesor poate pune mai multe note aceluiași elev.

Între entitățile **Profesor** și **Clasă** ar fi trebuit să existe o relație de many-to-many, deoarece un profesor poate preda la mai multe clase, iar la o clasă pot preda mai mulți profesori. De asemenea, între entitățile **Materie** si **Clasă** ar fi trebuit să existe o relație de many-to-many, deoarece o materie poate fi predată la mai multe clase, iar la o clasă se pot preda mai multe materii diferite. Astfel, pentru a nu mai crea două entități suplimentare ca urmare a transformărilor legăturilor many-to-many, a apărut tabela **Profesor – Clasa** care conține informații referitoare la ce profesor predă ce materie și la ce clase. În acest mod se explică existența relațiilor one-to-many dintre **Clasă, Materie, Profesor** și **Profesor-Clasă.**

Diagrama modelului logic

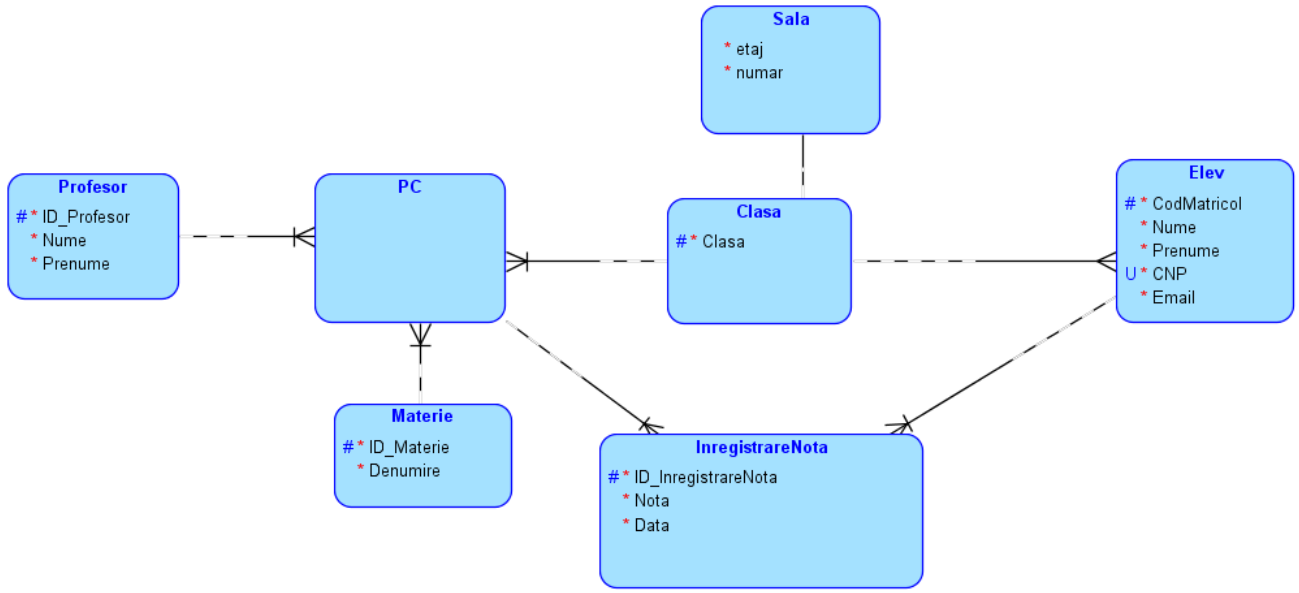
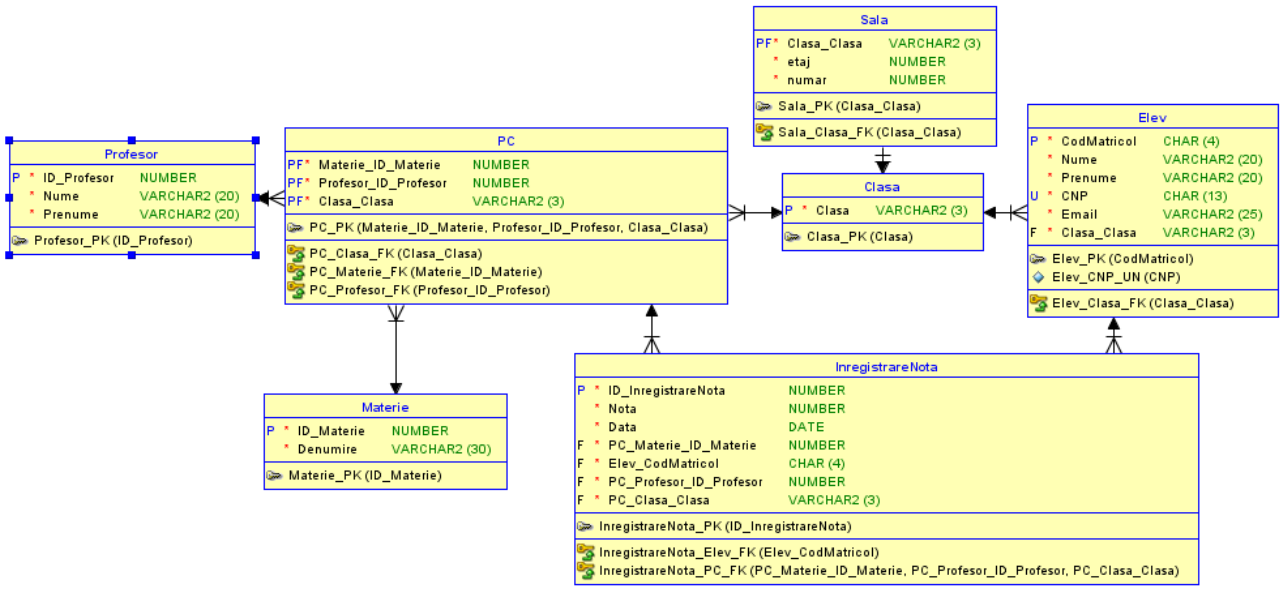


Diagrama modelului fizic

În modelul fizic am implementat auto-incrementări pentru cheile primare ale tabelelor Profesor, Materie și InregistrareNota. De asemenea am adăugat un trigger pe câmpul Data al tabelei InregistrareNota care verifică dacă data la care se încearcă să se treacă nota este mai mică sau egală cu data curentă și în același an cu data actuală.



Tehnologii folosite

Partea de back-end a fost realizată utilizând limbajul Python. Pentru partea de front-end s-a folosit pachetul tkinter, unul dintre cele mai utilizate pachete pentru implementarea interfetelor grafice în Python.

Conexiunea la baza de date

Pentru conectarea la baza de date s-a folosit funcția connect din modulul cx_Oracle, dându-i-se ca parametrii user-ul, parola și dsn-ul(data source name). În continuare se va obține un cursor folosindu-se metoda cursor a clasei conexiune returnată de funcția connect, urmând ca acest cursor să fie folosit pentru executarea comenzilor SQL. Programul stochează o conexiune la baza de date într-o variabilă globală din fișierul database.py, urmând ca acesta să fie importat în toate fișierele ce au nevoie de o conexiune. Conexiunea se va închide la încheierea programului folosind metoda connection.close()

Exemplu de cod:

```
import cx_Oracle

pwd = input('Introdu parola:')

con = cx_Oracle.connect(user="bd046",
                        password=pwd,
                        dsn="bd-dc.cs.tuiasi.ro:1539/orcl")
cur = con.cursor()
```

Interfața grafică

Actualizarea informațiilor unui elev

```
if self.varNume.get().isalpha() and self.varPrenume.get().isalpha() and len(self.varCNP.get()) == 13:
    print("UPDATE ELEV SET NUME = '{}', PRENUME = '{}', CNP = '{}', EMAIL = '{}', CLASA_CLASA = '{}'".format(
        self.varNume.get(), self.varPrenume.get(), self.varCNP.get(),
        self.varEmail.get(), self.varClasa.get()))

    cur.execute(
        "UPDATE ELEV SET NUME = '{}', PRENUME = '{}', CNP = '{}', EMAIL = '{}', CLASA_CLASA = '{}' WHERE "
        "CODMATRICOL = {}".format(
            self.varNume.get(), self.varPrenume.get(), self.varCNP.get(),
            self.varEmail.get(), self.varClasa.get(), self.codMatricol)
    )
    cur.execute("commit")
else:
    print('Unul dintre campuri nu este valid')
```

Inserarea unei noi note

```
rows = cur.execute("select id_materie from materie where denumire = '{}'".format(materie))

id_materie = ''
for row in rows:
    for elem in row:
        id_materie = elem

caractereInutile = "(,)'"
for caracter in caractereInutile:
    self.varProf.set(self.varProf.get().replace(caracter, ""))

id_profesor =''
rows = cur2.execute("SELECT id_profesor FROM PROFESOR WHERE nume || ' ' || prenume = '{}'"
                    .format(self.varProf.get()))
for row in rows:
    for elem in row:
        id_profesor = elem

cur2.execute("INSERT INTO INREGISTRARENOTA(nota,data,PC_Materie_ID_Materie, Elev_CodMatricol,PC_ID_Profesor,"
             "PC_Clasa_Clasa)  VALUES({}, TO_DATE('{}','fmDD-MM-YYYY'), {}, {}, {}, '{}')"
    .format(self.varNota.get(), self.varData.get(), id_materie, codMatricol, id_profesor, clasa))
cur2.execute("commit")
```

Calculul mediei generale a unui elev

```
rows2 = cur2.execute(
    "SELECT ROUND(avg(mediiMaterii),2) as Medie "
    "FROM (SELECT e.nume as Nume, e.prenume as Prenume, avg(nota) as mediiMaterii     "
        "from INREGISTRARENOTA i, elev e, pc, materie m "
        "where i.elev_codmatricol = e.codmatricol AND pc.materie_id_materie = m.id_materie AND "
        "pc.materie_id_materie = i.pc_materie_id_materie AND pc.profesor_id_profesor = i.PC_ID_Profesor AND "
        "pc.clasa_clasa = i.pc_clasa_clasa AND e.codmatricol = '{}' "
        "GROUP BY e.nume, e.prenume,m.denumire) "
    "GROUP BY nume, prenume"
        .format(result[0]))

medieGen = 0
for row in rows2:
    medieGen = row[0]
```

Exmatricularea unui elev(inclusiv ștergerea notelor)

```
def exmatriculeazaElev(self):
    codMatricol = self.master.additionalInfoFrames['CodMatricolElev']
    try:
        cur.execute('savepoint a')
        cur.execute("delete from inregistrarenota where elev_codmatricol = {}".format(codMatricol))
        cur.execute("delete from elev where codmatricol = {}".format(codMatricol))
        cur.execute("commit")
        self.master.switch_frame(elevi.Elevi)
    except cx_Oracle.Error as e:
        cur.execute('rollback to SAVEPOINT a')
        print(e)
```

Adăugarea unei clase și a sălii în care acea clasă va face orele

```
def submit(self):
    try:
        # e posibila situatia in care clasa sa fie valida insa sala sa fie ocupata, moment in care s-ar insera
        # o clasa fara sala
        cur.execute('savepoint a')
        cur.execute("INSERT INTO Clasa VALUES('{}')".format(self.varClasa.get()))
        cur.execute("INSERT INTO Sala VALUES('{}',{},{})".format(self.varClasa.get(), self.varEtaj.get(),
                                                                 self.varNrSala.get()))
        cur.execute('commit')
    except cx_Oracle.IntegrityError as e:
        cur.execute('rollback to SAVEPOINT a')
        print(e)
```