


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

DISCIPLINA: BAZE DE DATE

Gestiunea angajaților unei firme de construcții

Coordonator,
Prof. Mironeanu Cătălin

Student,
Budu Daniel
Grupa 1306B

IAȘI, 2022

1. Descrierea proiectului

Datorită numărului mare de angajați în cadrul unei firme de construcții este necesară prelucrarea și stocarea unei cantități mari de informații. Pentru rezolvarea acestei probleme am proiectat o aplicație care organizează datele intr-o bază de date care se poate gestiona cu ajutorul unei interfețe grafice.

2. Descrierea tehnologiilor folosite - Front-end

Aplicația a fost proiectată cu ajutorul limbajului **Python 3.11**, iar pentru partea grafică s-a folosit modulul **Tkinter**. Acest modul oferă o gamă largă de elemente grafice simpliste și ușor de folosit cu ajutorul căruia aplicația are un design intuitiv și ușor de folosit.

3. Descrierea tehnologiilor folosite - Back-end

Pentru partea de back-en am folosind **Python 3.11** și modulul **cx_Oracle**. In partea de back-end aplicația preia datele introduse de utilizator, le prelucrează și le incorporează intr-o comandă SQL corespunzătoare oprației dorite.

4. Structura tabelelor

Tabele accesibile prin interfață sunt:

- **Angajat**: sunt stocate date de interes ale angajaților firmei;
- **Departamente**: sunt stocate numele departamentelor unde lucrează angajații ;
- **Joburi**: sunt stocate informații despre fiecare meserie: intervalul de salarii, denumire meserie;
- **Locații**: sunt stocate informații referitoare la adresele departamentelor;

5. Inter-relaționarea entităților

În proiectarea bazei de date s-au identificat următoarele tipuri de relații: 1:1 (one-to-one), 1:n (one-to-many).

A) Relații 1:n

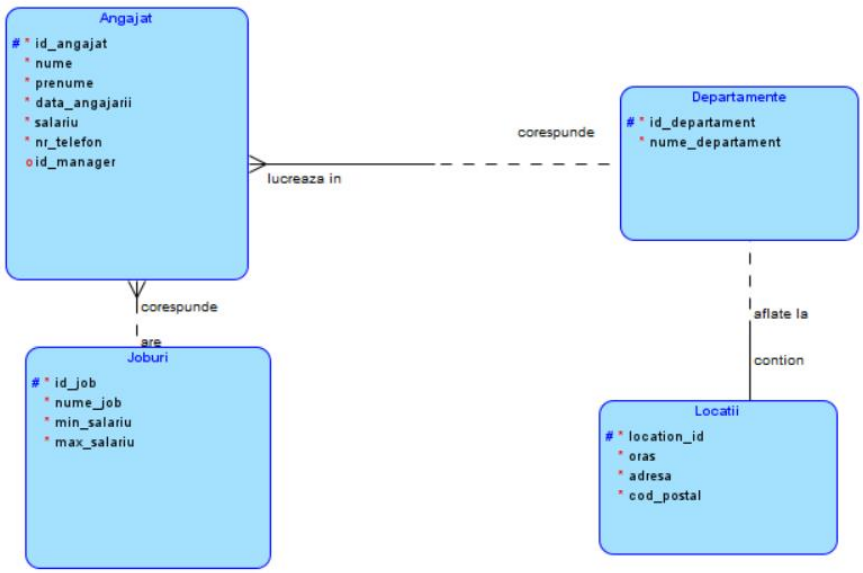
- Între entitatea **Joburi** și entitatea **Angajat** se stabilește o relație de 1:n. Unui anhajat ii este atribuit un singur job, dar mai mulți angajați pot practica acelasi job. Legătura între cele două entități se face prin atributul **ID_JOB.**

- Între entitatea **Departamente** și entitatea **Angajat** se stabilește o relație de 1:n. Asemanator cu relația anterioară, un angajat este asignat unui singur departament, dar in cadrul aceluiasi departament pot lucra mai mulți angajați. Legătura între cele două entități se face prin atributul **ID_DEPARTAMENT**.

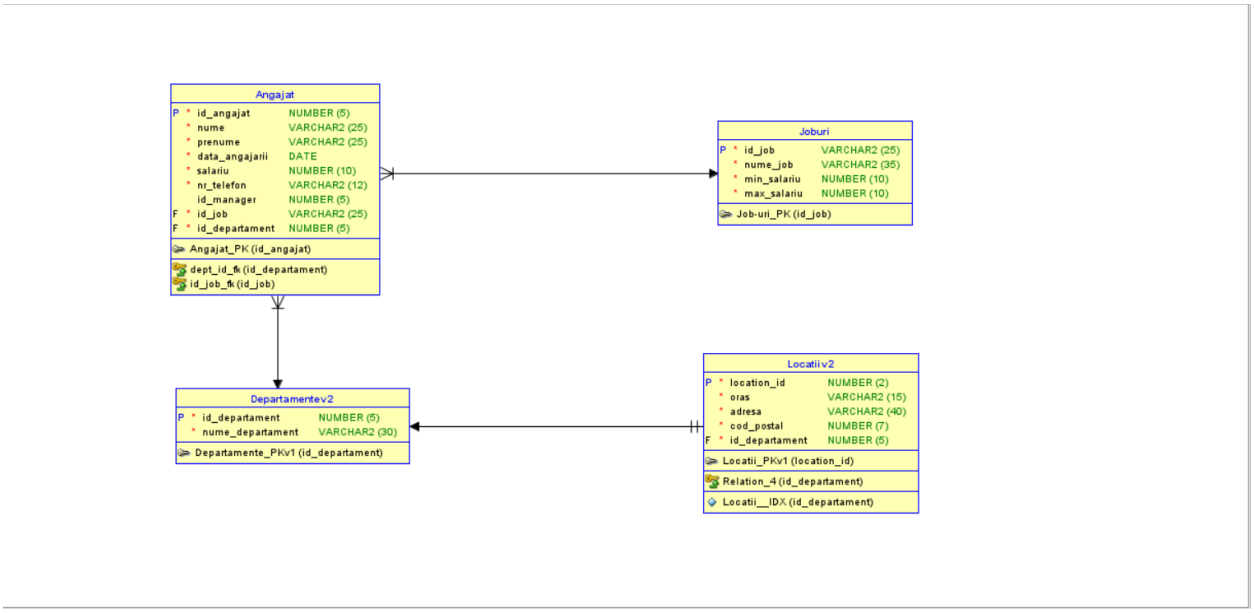
B) Relații 1:1

- Între entitatea **Departamente** și entitatea **Locații** se stabilește o relație de 1:1. Un departament are o locație unică, nici un departament nemaiputând avea aceeși locație. Legătura între cele două entități se face prin atributul **ID_DEPARTAMENT**.

Modelul logic:



Modelul relațional:



6. Aspecte legate de normalizare

Baza de date a fost normalizată, deoarece îndeplinește următoarele condiții:

A)Toate tabelele respectă condițiile primei forme normale:

- un atribut conține valori atomice din domeniul său (și nu grupuri de astfel de valori)
- nu conține grupuri care se repetă

B)Toate tabelele respectă condițiile celei de a doua forme normale:

- este prima formă normală
- toate atributele non-cheie depind de toate cheile candidat

C)Toate tabelele respectă a treia formă normală:

- este în a doua formă normală
- toate atributele non-cheie sunt direct (non-tranzitiv) dependente de toate cheile candidat.

7. Descrierea constrângerilor

In cadrul bazei de date am folosit constrângeri pentru a asigura corectitudinea datelor introduse,

Constrângerile de tip **CHECK** sunt folosite pentru: validarea datelor din coloanele **Oraș** și **Adresă** din tabela **Locații** și a coloanelor **Nume** și **Prenume** din tabela **Angajat** (datele trebuie să inceapă cu majusculă), datelor din coloana **NR_Telefon** din tabela **Angajat** (trebuie să fie format din 10 cifre), datelor din coloana **Cod_Postal** din tabela **Locații** (trebuie să fie format din 6 cifre).

Constrângerile de tip **NOT NULL** se găsesc pe coloanele ce conțin informații importante pentru înregistrări.

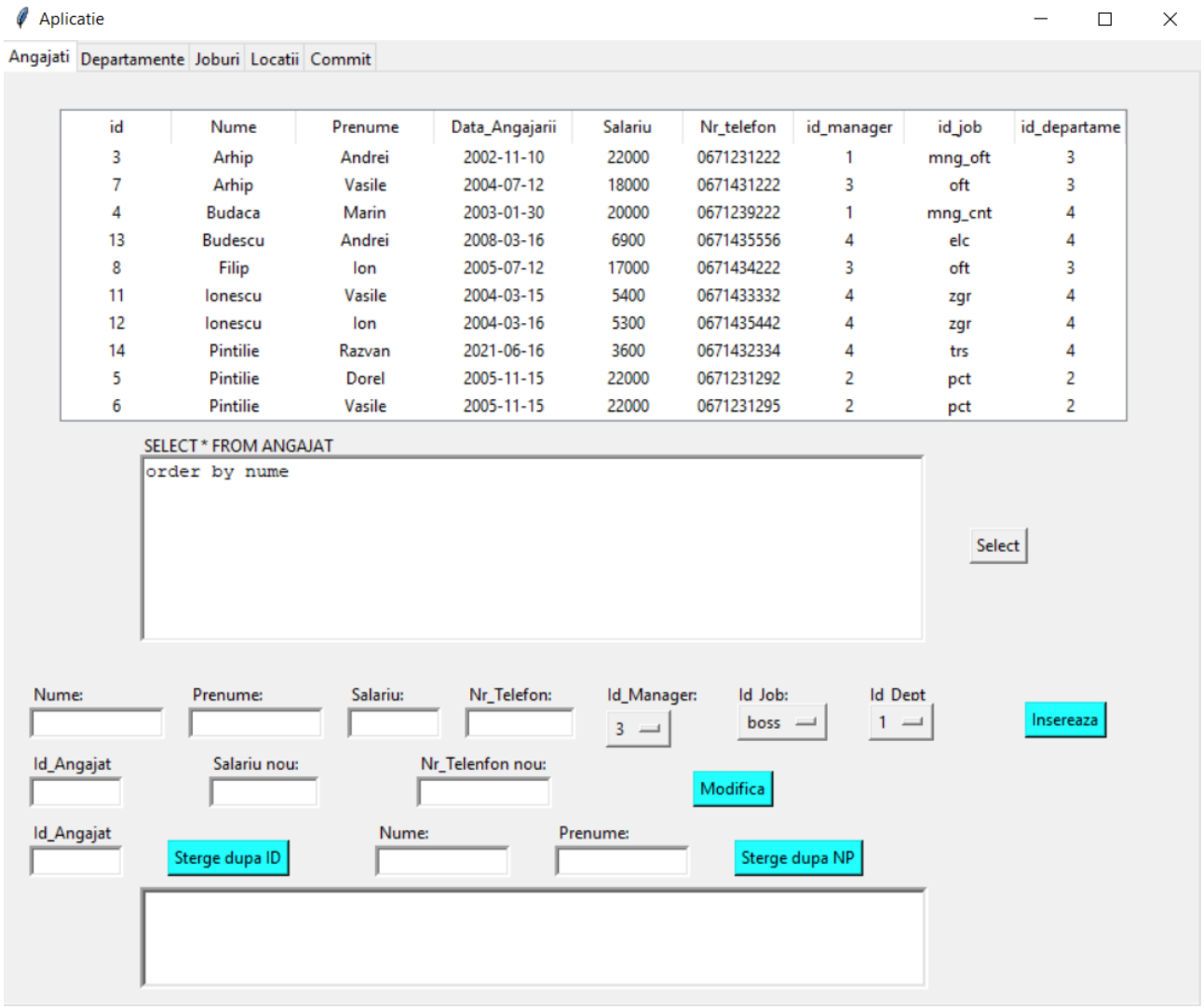
Primary key-urile sunt generate de baza de date printr-un mecanism de tip autoincrement, acestea neputând fi introduse din interfață.

8. Conectarea la baza de date

Conexiunea la baza de date este făcută prin intermediul modulului **cx_Oracle** din **Python 3.9**. Aceasta permite crearea unui obiect de tip conexiune cu care aplicația se conectează la baza de date. În momentul în care operația de conectare s-a realizat cu succes, vom putea crea un obiect de tip cursor cu ajutorul căruia vom putea execută comenzi specifice SQL precum : insert, update, delete, commit.

In cazul erorilor, aplicația afișează prin intermediul interfeței mesajul acestora pentru a avertiza utilizatorul.

9. Capturi de ecran cu interfața aplicației


Aplicatie
Angajati
Departamente
Joburi
Locatii
Commit
id Nume Prenume Data_Angajarii Salariu Nr_telefon id_manager id_job id_departame
3 Arhip Andrei 2002-11-10 22000 0671231222 1 mng_oft 3
7 Arhip Vasile 2004-07-12 18000 0671431222 3 oft 3
4 Budaca Marin 2003-01-30 20000 0671239222 1 mng_cnt 4
13 Budescu Andrei 2008-03-16 6900 0671435556 4 elc 4
8 Filip Ion 2005-07-12 17000 0671434222 3 oft 3
11 Ionescu Vasile 2004-03-15 5400 0671433332 4 zgr 4
12 Ionescu Ion 2004-03-16 5300 0671435442 4 zgr 4
14 Pintilie Razvan 2021-06-16 3600 0671432334 4 trs 4
5 Pintilie Dorel 2005-11-15 22000 0671231292 2 pct 2
6 Pintilie Vasile 2005-11-15 22000 0671231295 2 pct 2
SELECT * FROM ANGAJAT
order by nume
Select
Nume:
Prenume:
Salariu:
Nr_Telefon:
Id_Manager:
3
Id Job:
boss
Id Dept
1
Insereaza
Id_Angajat
Salariu nou:
Nr_Telenfon nou:
Modifica
Id_Angajat
Sterge dupa ID
Nume:
Prenume:
Sterge dupa NP

10. Capturi de ecran cu exemple de cod și instrucțiuni SQL folosite

```
# initialize the connection object
conn = None

try:
    # create a connection object
    conn = cx_Oracle.connect("bd019", "bd019", "bd-dc.cs.tuiasi.ro:1539/orcl")
    print(conn.version)


except Exception as err:
    print('Error while connecting to db')
    print(err)
```

```
def selFromAng(text):
    for item in angajati.get_children():
        angajati.delete(item)

    cur = conn.cursor()
    command = 'select * from angajat ' + text + ''
    try:
        cur.execute(command)
    except Exception as error:
        errAngajat.delete(1.0, END)
        errAngajat.insert(1.0, str(error))
    for result in cur:
        angajati.insert(parent='', index='end', text='',
                        values=(result[0], result[1], result[2], result[3].date(),
                                result[4], '0' + result[5], result[6], result[7], result[8]))
    cur.close()
```