UNIVERSITATEA TEHNICĂ “GH ASACHI” IAŞI

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE

SPECIALIZAREA: TEHNOLOGIA INFORMAŢIEI

DISCIPLINA: PROIECTAREA BAZELOR DE DATE

Gestionarea cursurilor electronice prin intermediul certificatelor intr-o companie

Coordonator,

Asist.dr.ing. Cătălin Mironeanu

Student,

Butnaru Silviu, 1409B

Iaşi, 2022

1. Descrierea proiectului;

Titlul proiectului: Gestionarea cursurilor electronice prin intermediul certificatelor intr-o companie.

In ceea ce priveste proiectul dezvoltat, se prezinta o abordare facila care usureaza procesul de atribuire asupra unui utilizator a unui curs/training, angajatul avand la dispozitie certificate digitale.

Fiecare utilizator are atribuit un numar de certificare electronice, cu o valabilitate de timp diferita, pe care le poate folosi pentru a accesa anumite cursuri.

Din perspectiva cursurilor, acestea au un numar limitat de accesari, o cantitate, care scade in functie de numarul de utilizator ce isi atribuie respectivul curs prin intermediul certificatului, care ulterior este distrus.

Asadar, utilizatorul ’w’ ofera certificatul ’x’ in schumbul cursului ’y’, certificatul este distrus, cantitatea cursului expus scade, apoi cursul ’y’ este atribuit utilizatorului ’w’.

2. Testere care pot fi rulate pentru a demonstra atingerea scopului propus;

Pentru a demonstra atingerea scopului propus, in scriptul cu teste avem un bloc anonim de cod ce efectueaza mai multe teste asupra tranzactiei principale, prezenta in pachetul „transaction_pack”, la rand, avand 9 teste efectuate in total. Testele **1, 2, 3, 5 si 7** testeaza atribuirea la un utilizator prin intermediul unui certificat a unui curs, rezultatul fiind unui pozitiv, de reusita; testele **4 si 8** testeaza esecul, utilizatorul neavand un certificat ce este oferit in schimbul unui curs; testele **6 si 9** testeaza esecul cand respectivul curs nu mai poate avea accesari/atribuiri, cantitatea acestuia este 0.

De asemenea, in acelasi fisier de testare mai exista un bloc anonim ce afiseaza continutul tuturor tabelelor, pentru o vizibilitate mai buna asupra procesarii de date executate in tranzactii.

In urma rularii blocului anonim de test vom primi urmatorul output:

// Tranzactia **1**

Tranzactie reusita!

// Tranzactia **2**

Tranzactie reusita!

// Tranzactia **3**

Tranzactie reusita!

// Tranzactia **4**

Tranzactie esuata! Certificat inexistent

// Tranzactia **5**

Tranzactie reusita!

//Tranzactia **6**

Insert sau update esuat! Cantitate 0

Tranzactie esuata! Cantitate 0

//Tranzactia **7**

Tranzactie reusita!

//Tranzactia **8**

Tranzactie esuata! Certificat inexistent

//Tranzactia **9**

Insert sau update esuat! Cantitate 0

Tranzactie esuata! Cantitate 0

Deci, se poate observa statusul fiecatui test al tranzactiei in parte si in cazul unei probleme, care a fost aceasta.

3. Structura si inter-relationarea tabelelor;

Tabelele prezente in aplicatie sunt:

- Employees;
- Certificates;
- Employee_certificate;
- Courses;
- Assigned_courses;

In aceasta arhitectura a bazei de date exista relatii de **1:1** cat si **M:N (1:N si N:1 din expandarea relatiei de M:N)**.

Relatia de tipul **1:1** este prezenta intre tabelele **employees** si **assignes_courses** deoarece un angajat isi poate atribui un curs o data si un curs este specific unui angajat. Legatura se face prin intermediul atributului **employees_id**.

De asemenea, intre tabelele **courses** si **assignes_courses** exista tot o relatie de **1:1** deoarece un curs este specific unui angajat iar un angajat iti poate atribui un curs specific o data. Legatura se face prin intermediul atributului **courses_id**.

Relatia de **M:N** este specifica tabelelor **employees** si **certificates** deoarece un angajat poate avea mai multe certificate iar mai multe certificate sunt specifice unui angajat.

Prin relatia de **M:N** se genereaza automat o tabela de legatura **employee_certificate**, in care exista relatia de **1:N** intre **employees** si **employee_certificate** alaturi de relatia **N:1** intre **employee_certificate** si **certificates**. Aceasta legatura se face prin intermediul atributelor **employees_id** si **certificates_id**.

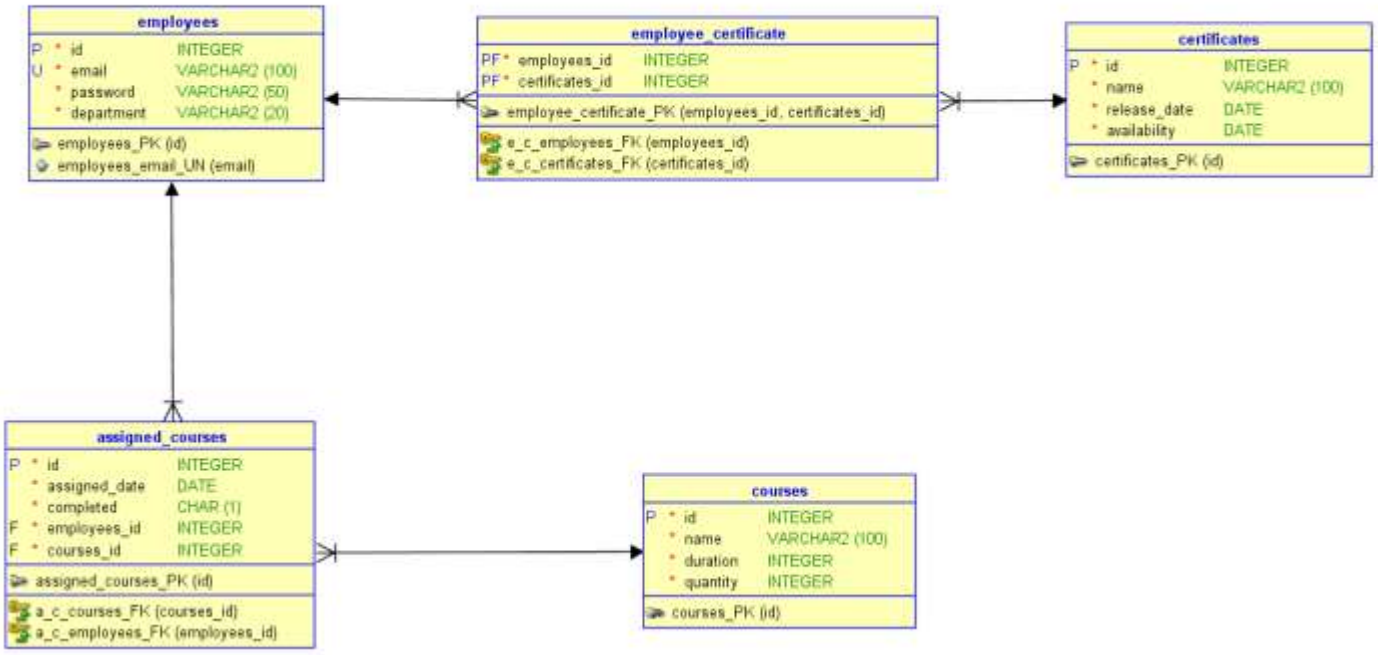

Diagrama Relationala

4. Descrierea logicii stocate;

In ceea ce priveste logica stocata in aplicatie, aceasta expune mai multe pachete, unele avand o interactiune comuna intre ele, acestea din urma continand o multitudine de proceduri, functii si exceptii exemplificand operatiile de tip CRUD (C-create, R-read, U-update, D-delete) ale stocarii persistente, alaturi de triggeri pentru o functionalitate corecta.

Triggeri:

- **employees_id_TRG**: *trigger* pentru iterarea automata la insert a id-ului in cadrul tabelei employees;
- **courses_id_TRG**: *trigger* pentru iterarea automata la insert a id-ului in cadrul tabelei courses;
- **certificates_id_TRG**: *trigger* pentru iterarea automata la insert a id-ului in cadrul tabelei certificates;
- **assigned_courses_id_TRG**: *trigger* pentru iterarea automata la insert a id-ului in cadrul tabelei assigned_courses;

- **check_quantity**: *trigger* ce verifica cantitatea din interiorul unui curs; in momentul in care se incearca o operatie de tip inser sau update si cantitatea este < 0, exceptia e_invalid_quantity este aruncata;

Pachete (proceduri, functii si exceptii):

- **crud_employees_pack**: *pachet* specific tabelei employees;
  - **create_employee**: *procedura* ce creaza o noua entitate de tip angajat; (C-create)
  - **read_first_specified_employees**: *procedura* ce intoarce un numar specific de angajati; (R-read)
  - **read_all_employees**: *procedura* ce intoarce toti angajatii; (R-read)
  - **read_employee_by_id**: *procedura* ce intoarce un angajat dupa un id; (R-read)
  - **read_employee_by_email**: *functie* ce primeste emailul unui angajat si intoarce id-ul acestuia; (R-read)
  - **update_employee_by_id**: *procedura* ce face update la un angajat; (U-update)
  - **delete_employee_by_id**: *procedura* ce sterge un angajat dupa un id; (D-delete)
  - **delete_all_employees**: *procedura* ce sterge toti angajatii; (D-delete)

- **crud_courses_pack**: *pachet* specific tabelei courses;
  - **create_course**: *procedura* ce creaza o noua entitate de tip curs; (C-create)
  - **read_first_specified_courses**: *procedura* ce intoarce un numar specific de cursuri; (R-read)
  - **read_all_courses**: *procedura* ce intoarce toate cursurile; (R-read)
  - **read_course_by_id**: *procedura* ce intoarce un curs dupa un id; (R-read)
  - **read_course_by_name**: *functie* ce primeste numele unui curs si intoarce id-ul acestuia; (R-read)
  - **read_course_quantity_by_id**: *functie* ce primeste un id de curs si intoarce cantitatea acestuia; (R-read)

- **update_course_by_id**: *procedura* ce face update la un curs; (U-update)
- **update_couse_quantity_by_id**: *procedura* ce face update la cantitatea dintr-un curs; (U-update)
- **delete_course_by_id**: *procedura* ce sterge un curs dupa un id; (D-delete)
- **delete_all_courses**: *procedura* ce sterge toate cursurile; (D-delete)

- **crud_certificates_pack**: *pachet* specific tabelei certificates;
  - **create_certificate**: *procedura* ce creaza o noua entitate de tip certificat; (C-create)
  - **read_first_specified_certificates**: *procedura* ce intoarce un numar specific de certificate; (R-read)
  - **read_all_certificates**: *procedura* ce intoarce toate certificatele; (R-read)
  - **read_certificates_by_id**: *procedura* ce intoarce un certificat dupa un id; (R-read)
  - **read_certificate_by_name**: *functie* ce primeste numele unui certificat si intoarce id-ul acestuia; (R-read)
  - **update_certificate_by_id**: *procedura* ce face update la un certificat; (U-update)
  - **delete_certificate_by_id**: *procedura* ce sterge un certificat dupa un id; (D-delete)
  - **delete_all_certificates**: *procedura* ce sterge toate certificatele; (D-delete)

- **crud_employee_certificate_pack**: *pachet* specific tabelei employee_certificate;
  - **create_employee_certificate**: *procedura* ce creaza o noua entitate de tip employee_certificate; (C-create)
  - **read_all_employee_certificate**: *procedura* ce intoarce toate entitatile de tipul employee_certificate; (R-read)
  - **delete_employee_certificate_by_employee_id_and_certificates_id**: *procedura* ce sterge un employee_certificate dupa un employee_id si un certificates_id; (D-delete)

- **delete_all_employee_certificate**: *procedura* ce sterge toate entitatile de tipul employee_certificate; (D-delete)

- **transaction_pack**: *pachet* specific tranzactiei realizate in proiect;
  - **e_invalid_quantity**: *exceptie* aruncata de triggerul **check_quantity** atunci cand cantitatea dintr-un curs este < 0;
  - **transaction_assigned_courses**: *procedura* ce creaza o tranzactie cu urmatorul flow: ca date de intrare, employee_email, course_name, certificate_name, pentru toate acestea se extrage id-ul fiecaruia (cu cate o functie specifica expusa in celelalte pachete mai sus explicate) alaturi de cantitatea unui curs (functie din crud_courses_pack). Se incearca scaderea cantitatii unui curs cu 1, autoapelandu-se triggerul **check_quantity**. Se verifica daca angajatul respectiv are sau nu certificate (functia check_existing_certificate privata pentru acest pachet) apoi se creaza o noua entitate in tabela assigned_courses cu id-urile aferente. La final se sterge certificatul angajatului respectiv. Daca nu exista nicio problema (nu se arunca nicio exceptie) tranzactia este realizata cu succes, se da commit. In cazul aruncarii unei exceptii un rollback este apelat, tranzactia nereusind deci nicio operatie de mai sus nu este executata;
  - **e_invalid_certificate**: *exceptie privata* pentru invaliditatea unui certificat;
  - **check_existing_certificate**: *procedura privata* ce verifica daca un utilizator are sau nu certificat; daca raspunsul este negativ, se arunca exceptia e_invalid_certificate;
  - **create_assigned_courses**: *procedura privata* ce insereaza o noua entitate in tabelul assigned_courses;
  - **read_all_assigned_courses**: *procedura* ce intoarce toate entitatile assigned_courses; (R-read)