PROIECT

ADMINISTRATREA BAZELOR DE DATE

ADMINISTRAREA UNUI SISTEM BANCAR

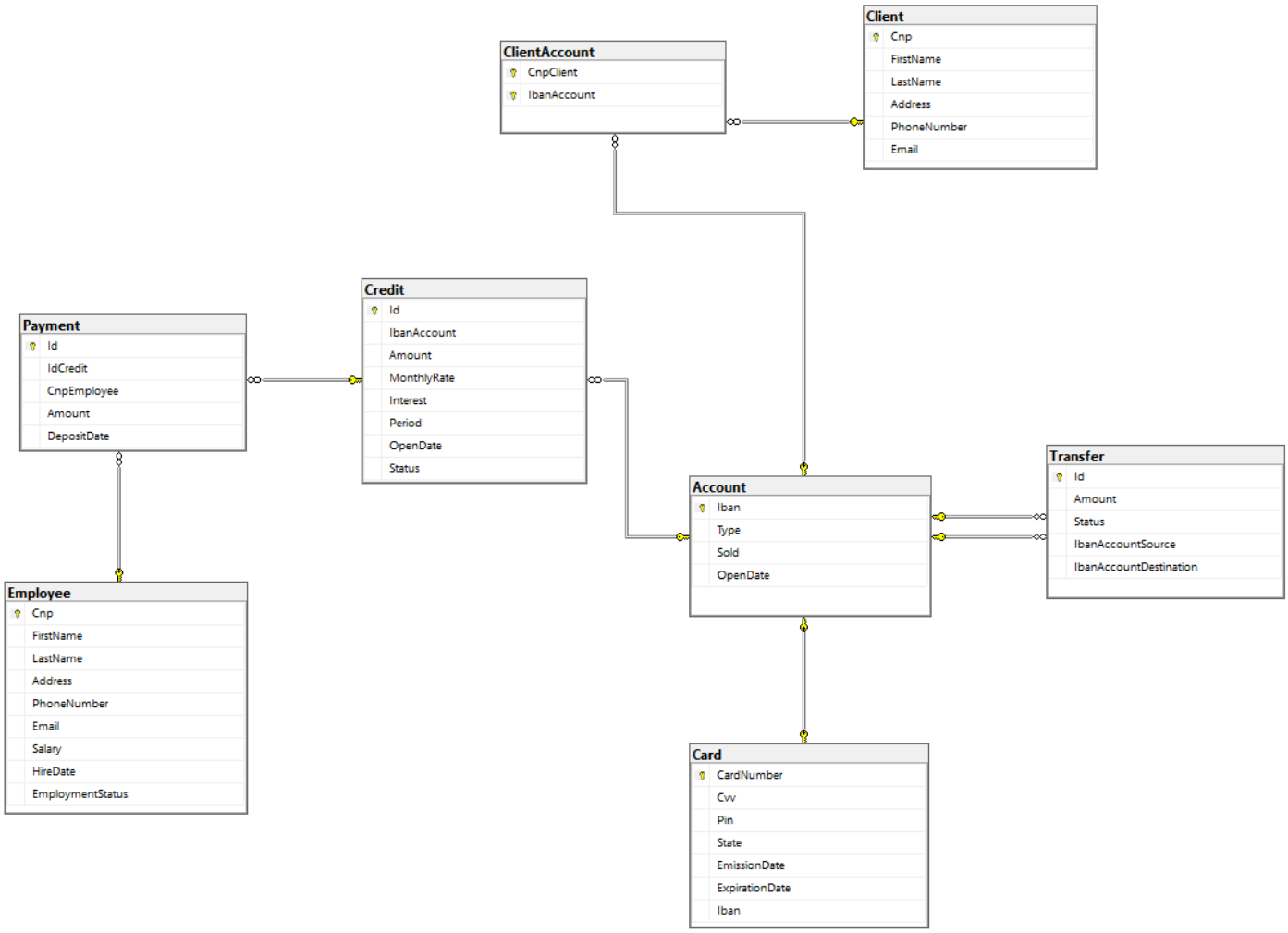
Grupa: 30641

Studenți: Irini Karina, Neghină Laurențiu

1. INTRODUCERE

Această lucrare prezintă un proiect care vizează dezvoltarea și administrarea unei baze de date pentru un sistem bancar. Proiectul se concentrează pe utilizarea tabelelor și a constrângerilor pentru a asigura integritatea și securitatea datelor. Baza de date este proiectată să reflecte principalele entități și relații dintr-o instituție financiară modernă. Printre acestea se numără clienții, conturile, cardurile, transferurile, angajații și creditele. De asemenea, proiectul integrează proceduri și funcții pentru automatizarea proceselor, joburi pentru execuția periodică a sarcinilor și mecanisme de backup și restaurare, asigurând protecția și recuperarea datelor. Se utilizează cursoare și triggere de tip DML și DDL pentru gestionarea actualizărilor și menținerea consistenței datelor. În plus, este prevăzută gestionarea utilizatorilor sistemului și a drepturilor lor de acces asupra elementelor bazei de date.

2. DIAGRAMA BAZEI DE DATE



Această diagramă ilustrează relațiile bazei de date relaționale proiectate.

1. **Client și Account**

Relația de tip M:N dintre aceste două entități este gestionată prin intermediul tabelei de legătură ClientAccount, sistemul fiind structurat astfel:

- Relație 1:N între tabelele Client și ClientAccount, deoarece un client poate avea mai multe conturi asociate.
- Relație 1:N între tabelele Account și ClientAccount, deoarece un cont poate fi asociat cu mai mulți clienți, în cazul conturilor de tip joint.

2. **Client și Card**

Relație de tip 1:N, deoarece un client poate deține mai multe carduri în cadrul aceleiași bănci.

3. **Account și Card**

Relație de tip 1:N, deoarece un cont poate avea asociate mai multe carduri bancare.

4. **Transfer și Account**

Relație de tip 1:N atât pentru IbanAccountSource, cât și pentru IbanAccountDestination, deoarece un cont poate genera mai multe transferuri ca sursă și poate primi numeroase transferuri.

5. **Account și Credit**

Relație de tip 1:N, întrucât un cont poate avea asociate mai multe credite în cadrul unei bănci.

6. **Credit și Payment**

Relație de tip 1:N, deoarece un credit poate avea mai multe plăți asociate.

7. **Employee și Payment**

Relație de tip 1:N, întrucât un angajat poate înregistra mai multe plăți.

3. VEDERI

- **AccountClients**

Aceasta adună informații despre conturile bancare și titularii acestora. Afișează IBAN-ul contului, tipul acestuia (individual sau comun), soldul curent, data deschiderii și lista titularilor. În cazul conturilor comune, sunt listate toate numele titularilor separate prin virgulă, iar pentru conturile individuale se afișează doar numele titularului principal.

- **TransferDetails**

Aceasta oferă detalii despre transferurile între conturi. Sunt incluse informații precum ID-ul transferului, suma transferată, statusul acestuia (“în curs” sau “finalizat”), IBAN-ul conturilor sursă și destinație, dar și listele de emailuri ale titularilor conturilor sursă și destinație. Aceste liste ajută la identificarea clară a persoanelor implicate în transferuri.

- **CreditPaymentSummary**

Aceasta sumarizează informațiile despre creditele active și plățile efectuate. Afișează ID-ul creditului, IBAN-ul contului asociat, suma creditului, rata lunară, perioada de rambursare,

data deschiderii și suma totală plătită până la momentul actual. Această vedere este utilă pentru a urmări situația financiară a creditelor active.

- **ClientCards**

Aceasta conține informații despre cardurile emise pentru clienți. În vedere sunt incluse detalii precum CNP-ul clientului, numele complet, numărul de telefon, detaliile cardului (numărul, data de expirare și starea acestuia), dar și informațiile contului bancar asociat (IBAN-ul și data deschiderii). Sunt afișate doar cardurile care sunt în prezent active.

- **EmployeePerformanceSummary**

Aceasta oferă o analiză a performanței angajaților în procesarea plăților. Sunt incluse date precum CNP-ul angajatului, numele complet, numărul total de plăți procesate și suma totală prelucrată de fiecare angajat. Acest rezumat ajută la evaluarea eficienței și productivității fiecărui angajat în gestionarea operațiunilor bancare.

4. PROCEDURI STOCATE

- **AddClient**

Această procedură adăuga un nou client în baza de date. Primește ca parametri CNP-ul, numele, prenumele, adresa, numărul de telefon și adresa de email ale clientului. Procedura inserează aceste informații în tabelul Client și confirmă adăugarea printr-un mesaj de succes.

- **UpdateCardState**

Această procedură actualizează starea unui card existent și este utilizată, de exemplu, atunci când un client dorește să își blocheze cardul. Procedura primește numărul cardului și noua stare dorită ca parametri, actualizează înregistrarea corespunzătoare din tabelul Card și informează utilizatorul că modificarea a fost realizată cu succes.

- **GetTransfersAsDestination**

Această procedură permite obținerea unei liste de transferuri în care un anumit client este destinația. Primește CNP-ul clientului ca parametru, iar apoi selectează transferurile relevante din tabelele Transfer, Account și ClientAccount. Transferurile sunt afișate în ordine descrescătoare după ID. Această metodă poate fi utilizată într-o aplicație reală pentru a oferi clienților posibilitatea de monitorizare a tranzacțiile efectuate către conturile lor.

- **DeleteExpiredCards**

Această procedură șterge toate cardurile expirate. Verifică cardurile cu o dată de expirare mai mică decât data curentă și le elimină din tabelul Card. Operația este însoțită de un mesaj care confirmă ștergerea. Un scenariu în care această metodă poate fi utilizată într-o aplicație reală este atunci când banca dorește să îndepărteze automat cardurile care nu mai sunt valabile.

- **ExecuteTransfer**

Această procedură procesează un transfer identificat prin ID. Procedura obține detalii despre suma transferată, conturile sursă și destinație și verifică dacă transferul este în starea “Pending”. Dacă există fonduri suficiente în contul sursă, transferul este realizat, starea este actualizată la “Completed”, iar sumele asociate conturilor sunt modificate. În caz contrar,

transferul eșuează, iar starea devine „Failed”. Operațiunea se efectuează într-o tranzacție pentru a asigura consistența datelor.

- **InsertTransfer**

Această procedură adaugă un transfer nou între două conturi. Primește suma, IBAN-ul contului sursă și IBAN-ul contului destinație. Procedura verifică dacă sursa și destinația sunt diferite, inserează transferul cu starea “Pending” și apelează procedura ExecuteTransfer pentru procesarea acestuia.

5. FUNCȚII

- **GetAccountBalanceByCardAndPin**

Această funcție returnează soldul contului asociat unui card, în funcție de numărul de card și PIN-ul introdus. Funcția verifică mai întâi dacă numărul de card există și dacă PIN-ul introdus este corect. Dacă da, returnează soldul contului asociat cardului.

- **GetAverageAnnualPayment**

Această funcție calculează valoarea medie anuală plătită pentru un cont bancar care are credite asociate. Funcția primește ca parametru IBAN-ul contului și returnează suma totală plătită pe parcursul anilor pentru creditele asociate acelui cont.

- **CalculateUpdatedSum**

Această funcție calculează suma actualizată a unui împrumut sau credit, utilizând formula dobânzii simple. Funcția primește suma inițială, dobânda anuală și perioada în ani și returnează suma actualizată care include dobânda aplicată.

- **CalculateMonthlyRate**

Această funcție calculează rata lunară a unui împrumut sau credit. Primește suma totală și perioada în ani, iar apoi returnează suma lunară care trebuie plătită, calculată prin împărțirea sumei totale la numărul total de luni.

- **GetTransfersByClient**

Această funcție oferă un set de transferuri efectuate din conturile unui client specificat prin CNP. Returnează ID-ul transferului, suma transferată, statusul transferului și IBAN-urile conturilor destinație

6. TRIGGERE DML ȘI DDL

- **trg_UpdateAmount_OnInsertCredit,**

Acest trigger DML actualizează suma și rata lunară a unui credit după inserarea unei noi înregistrări. După ce se obțin valorile inițiale ale creditului, trigger-ul calculează suma actualizată și rata lunară folosind funcțiile corespunzătoare, apoi actualizează aceleași câmpuri în tabelul Credit.

- **trg_UpdateCreditStatus**

Acest trigger DML actualizează statusul unui credit după ce a fost înregistrată o plată în tabelul Payment. Dacă suma totală plătită pentru un credit este mai mare sau egală cu valoarea totală a creditului, trigger-ul schimbă statusul acestuia în “Completed”.

- **trg_DeleteAccount,**

Acest trigger DML previne ștergerea unui cont dacă există credite nefinalizate sau transferuri în așteptare. Dacă aceste condiții nu sunt îndeplinite, trigger-ul șterge contul și toate datele asociate (credite, transferuri, plăți, carduri).

- **trg_LogDDLChanges**

Acest trigger DDL loghează modificările de tip CREATE_TABLE, ALTER_TABLE, DROP_TABLE într-o bază de date. Când se face o modificare a unui tabel, se salvează detalii precum tipul evenimentului, numele tabelului și momentul acțiunii în tabelul ChangesLog, oferind un istoric al modificărilor structurale ale bazei de date.

- **trg_PreventDropSensitiveTables**

Acest trigger DDL previne ștergerea tabelelor sensibile, cum ar fi Credit și Account. Dacă se încearcă ștergerea unui astfel de tabel, trigger-ul blochează acțiunea și afișează o eroare, protejând astfel datele importante.

7. CURSOARE

- **ClientAccountCursor**

Acest cursor parcurge tabela Client și tabelul asociat ClientAccount, obținând informații despre CNP-ul clientului, numele complet, soldul și tipul contului. Fiecare înregistrare este afișată cu detalii despre client și soldul asociat contului. Această operațiune poate fi folosită pentru a face audite ale soldurilor clientului, identificând conturile cu solduri neobișnuite sau eronate care necesită verificare.

- **CreditCursor**

Acest cursor este folosit pentru a parcurge creditele active din tabela Credit. Fiecare primește o creștere a dobânzii, iar suma creditului și rata lunară sunt actualizate pe baza noii dobânzi. Calculul este realizat prin apelarea unor funcții definite în baza de date. Poate fi folosit pentru recalcularea automată a dobânzilor pentru clienții care au credite pe termen lung și care sunt afectați de fluctuațiile dobânzii.

- **SalaryCursor**

Acest cursor parcurge tabela Employee pentru angajații activi și actualizează salariile acestora, aplicând o creștere procentuală determinată de rata inflației. Salariile sunt actualizate direct în tabela Employee. Este util într-un sistem de salarizare automatizat care aplică ajustări periodice ale salariilor angajaților băncii pentru a compensa efectele inflației.

- **RejectedCreditCursor**

Acest cursor parcurge tabela Credit și șterge toate creditele care au statusul Rejected. Fiecare credit cu acest status este eliminat din baza de date. Această operațiune poate fi utilizată

pentru a curăța periodic baza de date, îndepărtând creditele respinse, ceea ce contribuie la îmbunătățirea performanței sistemului.

- **ClientCursor**

Acest cursor combină informații din tabelele Client, ClientAccount, Account, Card și Credit, afișând detalii complete despre clienți, conturile acestora, cardurile asociate și creditele existente. Detaliile sunt prezentate pentru fiecare client în parte. Poate fi folosit pentru a crea rapoarte financiare periodice ale clienților sau pentru a oferi suport în analiza comportamentului acestora, identificând tipare de utilizare a creditelor, cardurilor și a conturilor bancare, îmbunătățind serviciile personalizate

8. UTILIZATORI

Se creează login-uri și utilizatorilor pentru diferite roluri:

- **Administrator:** Se creează un login pentru utilizatorul Administrator cu o parolă specifică și un utilizator asociat în baza de date. Acesta primește permisiuni complete asupra bazei de date (GRANT CONTROL), adică are control deplin asupra tuturor obiectelor și datelor din baza de date.
- **Employee:** Se creează un login pentru utilizatorul Employee și un utilizator asociat în baza de date. Angajatul primește permisiuni de execuție asupra unor procedure specifice, cum ar fi AddClient și DeleteExpiredCards, pentru a putea executa aceste acțiuni în baza de date, dar fără a avea acces complet la datele sensibile.
- **Client:** Se creează un login pentru utilizatorul Client și un utilizator asociat în baza de date. Acesta primește permisiuni de execuție asupra unor proceduri și funcții care permit accesul la datele sale, de exemplu, pentru obținerea soldului contului folosind un card și PIN, sau pentru vizualizarea transferurilor ca destinatar.

9. JOB-URI SQL

- **CountPaymentsJob**

Jobul numără plățile efectuate în ziua precedentă și le salvează într-un tabel de loguri, PaymentLog. Un pas denumit CountPaymentsStep rulează un cod T-SQL care obține data zilei anterioare și numără plățile din tabela Payment, inserând rezultatul în PaymentLog. Jobul este programat să ruleze zilnic la miezul nopții și este atașat unui program care îl pornește automat.

- **BackupTablesJob**

Jobul creează backup-uri complete pentru tabelele esențiale asigurându-se că datele sunt salvate periodic. Un pas denumit BackupStep execută un cod T-SQL care definește un director pentru backup și creează backup-uri pentru tabelele Client, Account, Employee, Credit și Payment, folosind data curentă în numele fișierelor. Jobul este programat să ruleze zilnic la ora 02:00 AM și este atașat unui program care îl pornește automat.

10. BACKUP ȘI RESTORE

- **Backup complet**

Se creează un backup complet, care cuprinde toate datele și structurile bazei de date. Fișierul este salvat în locația specificată, iar opțiunea WITH FORMAT asigură formatarea fișierului înainte de a începe procesul, eliminând orice fișier existent. Prin INIT, se permite suprascrierea fișierului anterior, iar opțiunea NAME atribuie un nume descriptiv backup-ului. Progresul procesului de backup este urmărit și raportat la fiecare 10% din finalizarea acestuia, utilizând STATS = 10.

- **Backup diferențial**

Se creează un backup diferențial, care salvează doar modificările apărute de la ultimul backup complet. Acest tip de backup este mai rapid și eficient, deoarece doar datele noi sau modificate sunt incluse. Similar cu backup-ul complet, opțiunea INIT permite suprascrierea fișierului existent de backup, iar progresul procesului este monitorizat prin STATS = 10, care afișează procentul de finalizare.

- **Restore**

Restaurarea bazei de date se realizează prin două comenzi distincte. Prima comandă restaurează întreaga bază de date din backup-ul complet, făcând-o accesibilă pentru utilizare. A doua comandă restaurează modificările efectuate de la ultimul backup complet, folosind backup-ul diferențial. În ambele situații, opțiunea WITH RECOVERY finalizează procesul și face baza de date disponibilă pentru utilizatori.