UNIVERSITATEA TEHNICA “Gheorghe Asachi” din IAȘI

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

DOMENIUL: Calculatoare și Tehnologia Informației

SPECIALIZAREA: Tehnologia Informației

DISCIPLINA: PROIECTAREA BAZELOR DE DATE

Sistem de Management pentru Cinema

Coordonator,

Prof. Cătălin Mironeanu

Student,

Mertic Lucia-Maria,

Grupa 1409A

1.Descrierea proiectului

Acest proiect are ca scop dezvoltarea unui sistem de management pentru un cinema. Sistemul va fi capabil să gestioneze diverse aspecte ale funcționării unui cinema, cum ar fi gestionarea informațiilor despre clienți, proiecțiile de filme și rezervările de bilete.
Gestionarea clienților: Sistemul va păstra o evidență a detaliilor clienților, cum ar fi numele, numărul de telefon și adresa de e-mail, facilitând astfel comunicarea dintre personalul cinema-ului și clienți.
Gestionarea filmelor: Sistemul va permite stocarea și gestionarea informațiilor despre filmele care rulează în cinematograf. Aceste informații includ numele filmului, durata și prețul, precum și detalii suplimentare: anul de apariție, regizorul și genul filmului.
Gestionarea programărilor: Sistemul va ține evidența programărilor de filme, inclusiv datele și orele de începere și finalizare, precum și numărul de locuri disponibile pentru fiecare proiecție.
Gestionarea rezervărilor: Sistemul va permite clienților să facă rezervări pentru proiecțiile de filme dorite.

2.Structura si inter-relationarea tabelelor

Modelul relational arata in felul urmator:
Avem relatie de 1:1 intre tabelele filme si detalii_film.
Avem relatie de 1:n intre tabelele filme si programari, intrucat un film poate fi difuzat de mai multe ori la un cinema.
Avem relatie de 1:n intre tabelele programari si rezervari, intrucat pentru difuzarea unui film pot fi facute multe rezervari de catre clienti.
Avem relatie de 1:n intre clienti si rezervari, intrucat un client poate face mai multe rezervari pentru diferite filme.

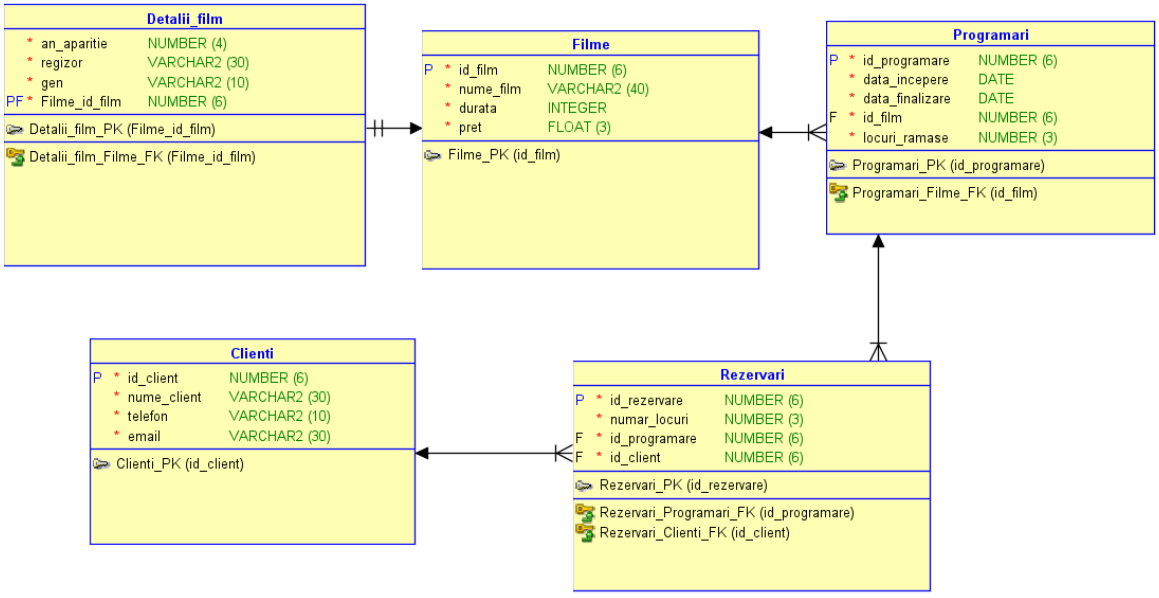

Tabela clienti:
id_client - NUMERIC: Cheia primară, valoare unică pentru fiecare client.
nume_client - VARCHAR2(30): Numele clientului, un șir de caractere.
telefon - VARCHAR2(10): Numărul de telefon al clientului, un șir de 10 caractere.
email - VARCHAR2(30): Adresa de email a clientului, un șir de caractere.
Avem constrângeri pentru numarul de telefon să fie fix de 10 cifre, si constrangere de validare pentru email cu un reggex.

Tabela filme:
id_film - NUMERIC: Cheia primară, valoare unică pentru fiecare film.
nume_film - VARCHAR2(40): Numele filmului, un șir de caractere.
durata - INTEGER: Durata filmului în minute.
pret - FLOAT: Prețul biletului pentru film.
Avem constrângeri ca durata și pretul biletului să fie numere pozitive.

Tabela detalii_film:
an_aparitie - NUMERIC: Anul apariției filmului.
regizor - VARCHAR2(30): Numele regizorului filmului, un șir de caractere.
gen - VARCHAR2(10): Genul filmului, un șir de caractere.
filme_id_film - NUMERIC: O cheie străină care este de asemenea si o cheie primara care face referire la id_film din tabela filme.
Avem contrângerea ca an_aparitie sa fie un numar intre 1800 si 2022. O altă constrangere este faptul ca genul filmului poate lua valori dintr-un enum impus.

Tabela programari:
id_programare - NUMERIC: Cheia primară, valoare unică pentru fiecare programare.
data_incepere - DATE: Data și ora la care începe proiecția filmului.
data_finalizare - DATE: Data și ora la care se încheie proiecția filmului.
id_film - NUMERIC: O cheie străină care face referire la id_film din tabela filme.
locuri_ramase - NUMERIC: Numărul de locuri rămase disponibile pentru o anumită proiecție.
Există două triggere pe această tabelă: trg_programari_data_finalizare și trg_programari_data_incepere. Aceste triggeri verifică dacă datele de început și sfârșit ale proiecțiilor sunt valide. Data_incepere>Sysdate, iar data_finalizare trebuie sa fie mai mare decat data_incepere.
Avem constrangere ca locuri_ramase sa fie >=0.

Tabela rezervari:
id_rezervare - NUMERIC: Cheia primară, valoare unică pentru fiecare rezervare.
numar_locuri - NUMERIC: Numărul de locuri rezervate.
id_programare - NUMERIC: O cheie străină care face referire la id_programare din tabela programari.
id_client - NUMERIC: O cheie străină care face referire la id_client din tabela clienti.
Avem constrangere ca numar_locuri sa fie >=1, deoarece nu ar avea sens ca un client sa faca o rezervare de 0 locuri la un film.

3.Testele care pot fi rulate pentru a demonstra atingerea scopului propus

Testarea sistemului de gestionare a unui cinema este un proces necesar pentru a ne asigura că fiecare funcționalitate funcționează corect. Iată o detaliere a fiecărei secțiuni din scriptul de testare:

1. Testare inserare: Am creat pachete de inserare pentru a adăuga înregistrări noi în tabele (clienti, filme, detalii_film, programari, rezervari). Pentru fiecare tabel, inserez câteva înregistrări noi și verific dacă acestea apar în baza de date.
De exemplu, pentru tabela "clienti", inserez clientul 'Lucia Mertic' cu numărul de telefon '0760558400' și adresa de email 'lucia_mertic@gmail.com'.

2. Testare update: Am creat pachete de actualizare pentru a modifica câmpuri pentru înregistrările existente în tabele.
De exemplu, în tabela "clienti", modific numele clientului cu id-ul 1 în 'Updated Client', numărul de telefon în '0999888777', și adresa de email în 'updated.client@email.com'.

3. Testare afișare: După operațiile de inserare și update, rulez proceduri pentru afișarea datelor din fiecare tabela pentru a verifica dacă conținutul tabelei reflectă modificările făcute.
De exemplu, pentru a verifica dacă modificările în tabela "clienti" au avut loc, rulez procedura 'print_package.print_client'.

4. Testare ștergere: Am creat scripturi de ștergere în funcție de id pentru fiecare tabela în parte.
De exemplu, pentru a șterge clientul cu id-ul 4, rulez procedura 'delete_package.delete_client(4)'.

Testare tranzacție:

Acestea sunt cele două proceduri explicite:

Procedura attempt_rezervare: Aceasta procedură încearcă să realizeze o rezervare în baza de date. În primul rând, verifică dacă există suficiente locuri pentru rezervare în tabela programari pentru o anumită proiecție de film (id_programare). Dacă există suficiente locuri, se actualizeaza numărul de locuri rămase în tabela programari (prin trigger). În caz contrar, afișează un mesaj de eroare, indicând că rezervarea nu poate fi realizată deoarece nu există suficiente locuri. De asemenea, dacă id-ul programării nu există în tabela programari, afișează un mesaj de eroare.

Procedura process_rezervare: Această procedură gestionează procesul de inserare, actualizare și ștergere a unei rezervări. În primul rând, obține id-ul maxim curent de la rezervari și îl incrementază cu 1 pentru a obține noul id de rezervare. Apoi, afișează toate înregistrările din programari și rezervari înainte de a face o nouă rezervare. În continuare, efectuează o inserare în tabela rezervari. După inserare, afișează din nou toate înregistrările din programari și rezervari. Ulterior, efectuează o operațiune de actualizare a numărului de locuri pentru noua rezervare și afișează din nou înregistrările. În final, șterge rezervarea pe care a introdus-o și afișează din nou înregistrările. Această procedură arată deci cum o rezervare este inserată, actualizată și ștearsă în baza de date și cum aceste operațiuni afectează înregistrările din tabelele programari și rezervari.

Aceste două proceduri sunt legate de tranzacția de rezervare. Prima încearcă să facă o rezervare, iar a doua gestionează întregul proces de rezervare. Ele ajută la asigurarea integrității datelor în baza de date, prin verificarea disponibilității locurilor înainte de a face o rezervare și prin afișarea datelor înainte și după fiecare operațiune de modificare a datelor.

4.Descrierea logicii stocate

Pachete: am utilizat pachete pentru a organiza logica de aplicație în module logice și pentru a gestiona procedurile și variabilele globale. Pachetele pot îmbunătăți performanța, gestionarea securității și pot permite reutilizarea și împărțirea codului în cadrul aplicației.
Pachete existente sunt: insert_package, update_package, delete_package, print_package, test_package (care contine cele doua proceduri ce testeaza functionarea corecta a tranzactiei).

Proceduri: Procedurile au fost utilizate pentru a executa operațiuni specifice asupra datelor din tabele, cum ar fi inserarea, vizualizarea, actualizarea și ștergerea datelor. Procedurile pot fi invocate de alte proceduri sau blocuri PL/SQL și pot accepta parametri de intrare și ieșire.

Triggere: Triggerele sunt utile în PL/SQL pentru a răspunde la evenimente specifice, cum ar fi inserarea, actualizarea sau ștergerea unei înregistrări într-o tabelă. Ele pot fi folosite pentru a impune reguli de afaceri complexe sau pentru a asigura integritatea datelor.

Primul trigger este un trigger numit **trigger_locuri_disponibile** ce functioneaza dupa BEFORE INSERT, UPDATE, DELETE pe tabela rezervari. Acest trigger este folosit pentru a menține acuratețea numărului de locuri rămase pentru fiecare programare în tabela programari. Dacă un rând este șters din rezervari, triggerul actualizează numărul de locuri rămase prin adăugarea numărului de locuri rezervate la numărul de locuri rămase. Dacă un nou rând este inserat, numărul de locuri rezervate este scăzut din numărul de locuri rămase. Dacă un rând este actualizat, triggerul calculează diferența dintre numărul de locuri vechi și cel nou și actualizează numărul de locuri rămase în mod corespunzător.

Al doilea trigger este un trigger numit **trg_programari_data_finalizare** ce functioneza dupa BEFORE INSERT OR UPDATE pe tabela programari. Acesta se asigură că data de finalizare a programării este ulterioară datei de începere. Dacă nu, triggerul aruncă o excepție și afișează un mesaj de eroare.

Al treilea trigger este, de asemenea, un trigger BEFORE INSERT OR UPDATE numit **trg_programari_data_incepere** pe tabela programari. Acesta se asigură că data de începere a programării este ulterioară datei curente. Dacă nu, triggerul aruncă o excepție și afișează un mesaj de eroare.

Excepții: Au fost utilizate pentru gestionarea erorilor în blocurile PL/SQL. În acest caz, dacă nu există date pentru un anumit id_programare, se va arunca excepția NO_DATA_FOUND, iar utilizatorul va fi înștiințat prin intermediul unui mesaj de eroare. (in procedura process_rezervare).

Cursori: Cursorii permit manipularea seturilor de rânduri returnate de o interogare SQL. În acest cod, au fost folosite pentru a parcurge și afișa rezultatele interogărilor SQL în cadrul procedurilor (print_package).