UNIVERSITATEA TEHNICĂ „Gheorghe Asachi” din IAȘI
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DOMENIUL: Calculatoare și Tehnologia Informației
SPECIALIZAREA: Tehnologia Informației

BAZA DE DATE PENTRU GESTIONAREA UNUI MAGAZIN DE JOCURI

Coordonator:
Mironeanu Cătălin

Student:
Petrișor Eduard-Gabriel
Grupa: 1408B

Iași, 2025

Contents

1 Descrierea Proiectului

Scopul acestui proiect este dezvoltarea unei baze de date comprehensive pentru gestionarea eficientă a unui magazin de jocuri video. Sistemul permite managementul complet al stocului de produse, evidența clienților, procesarea comenzilor și administrarea informațiilor despre publisheri.

Baza de date implementează un model complet de e-commerce pentru industria jocurilor, oferind funcționalități pentru:

- Gestionarea inventarului de jocuri cu informații detaliate despre preț, stoc și date de lansare
- Administrarea conturilor de clienți cu informații de contact și drepturi de acces
- Procesarea comenzilor cu verificări automate ale stocului disponibil
- Evidența publisherilor și asocierea acestora cu jocurile publicate
- Implementarea de măsuri de securitate și validare a datelor

2 Teste Realizate și Tranzacții

Pentru demonstrarea funcționalității complete a sistemului, au fost implementate teste care acoperă toate operațiunile CRUD și scenariile de eroare posibile.

2.1 Teste pentru Funcționalitățile de Bază (CRUD)

CREATE:

- Adăugarea de noi clienți impreuna cu informatiile lor de contact
- Inserarea de jocuri noi
- Plasarea de comenzi

READ:

- Interogarea informațiilor despre clienți și datele lor de contact
- Interogarea catalogului de jocuri
- Vizualizarea istoricului de comenzi pentru fiecare client

UPDATE:

- Actualizarea stocurilor de jocuri
- Modificarea prețurilor produselor
- Editarea informațiilor de contact ale clienților

DELETE:

- Anularea comenzilor cu restaurarea automată a stocului
- Eliminarea jocurilor din catalog
- Ștergerea conturilor de clienți

2.2 Tranzacții Implementate

Toate procedurile implementate sunt încapsulate în tranzacții cu puncte de salvare (SAVEPOINT), asigurând consistența datelor și posibilitatea de rollback în caz de eroare. Tranzacțiile principale includ plasarea comenzilor, actualizarea stocurilor și gestionarea informațiilor clienților.

3 Structura și Inter-relațiile Tabelelor

3.1 Structura

Baza de date este structurată în jurul a cinci tabele, organizate într-un model relațional normalizat:

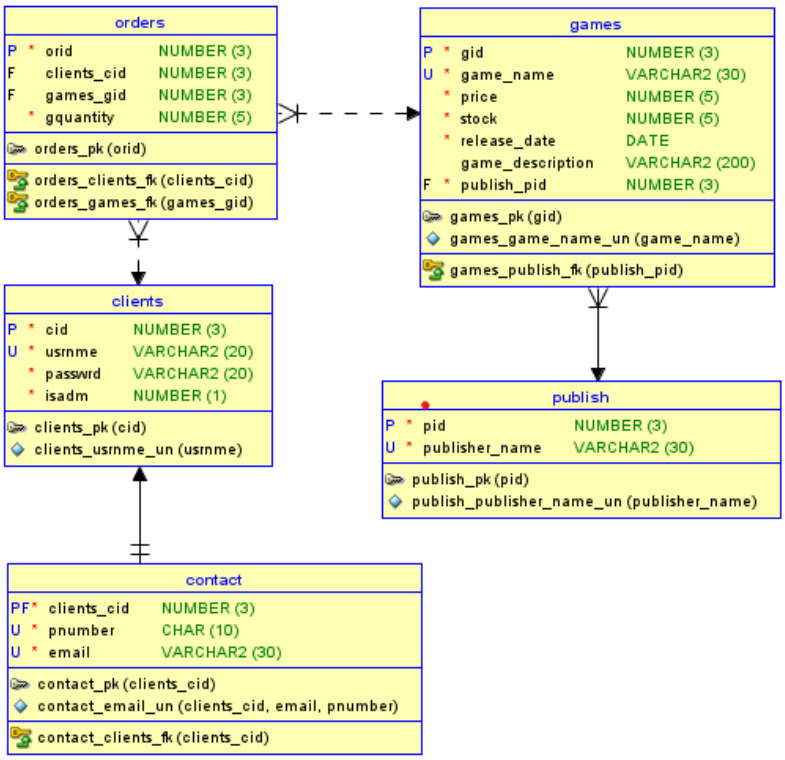

Figure 1: Structura Tabelelor.

3.2 Tipurile de Relații Implementate

One-to-One (1:1):

- CLIENTS → CONTACT: Fiecare client are exact un set de informații de contact. Această separare permite o mai bună organizare a datelor și optimizarea interogărilor care nu necesită informații de contact.

One-to-Many (1:N):

- CLIENTS → ORDERS: Un client poate plasa multiple comenzi
- GAMES → ORDERS: Un joc poate fi comandat de mulți clienți
- PUBLISH → GAMES: Un publisher poate publica multiple jocuri

Many-to-Many (M:N):

- CLIENTS → GAMES: Prin intermediul tabelei ORDERS,

3.3 Constrângeri și Validări

Sunt implementate constrângeri și validări pentru:

- Validarea formatului numerelor de telefon (07XXXXXXXX)
- Validarea formatului adreselor de email
- Validarea formatului datelor de lansare
- Unicitatea unor câmpuri cum ar fi: Games.GAME_NAME, Publish. PUBLISHER_NAME, Clients.USRNME etc.
- Asigurarea ca datele credidentiale sunt compuse din caractere alfanumerice

- Prevenirea introducerii de randuri cu date lipsă în câmpuri esențiale precum Contact.PNUMBER, Games.PRICE, Orders.GQUANTITY.
- Prevenirea introducerii sau modificarii comenzilor, prin intermediul unui Trigger, în cazul în care cantitatea solicitată depășește stocul disponibil al produsului.

4 Descrierea Logicii Stocate

4.1 Secvențe (Sequences)

Pentru generarea automată a cheilor primare, sistemul utilizează patru secvențe:

- SEQ_CLIENTS: Pentru identificatorii unici ai clienților
- SEQ_GAMES: Pentru identificatorii jocurilor
- SEQ_ORDERS: Pentru identificatorii comenzilor
- SEQ_PUBLISH: Pentru identificatorii publisherilor

4.2 Triggeri (Triggers)

Triggeri pentru Auto-incrementare:

- TRG_CLIENTS_BI: Atribuie automat următorul ID din secvență la inserarea unui client nou
- TRG_GAMES_BI: Atribuie automat următorul ID din secvență la inserarea unui joc nou
- TRG_ORDERS_BI: Atribuie automat următorul ID din secvență la inserarea unei comenzi noi
- TRG_PUBLISH_BI: Atribuie automat următorul ID din secvență la inserarea unui publisher nou

Trigger pentru Validarea Stocului:

- TRG_ORDERS_STOCK_CHECK este un trigger care se execută inainte de inserare sau actualizare pe tabela ORDERS. Acesta verifică automat dacă cantitatea comandată nu depășește stocul disponibil, ridicând o excepție personalizată în caz contrar.

Tipuri de date PL/SQL utilizate în triggeri:

- GAMES.STOCK%TYPE pentru menținerea consistenței tipurilor de date
- Utilizarea :NEW pentru accesarea valorilor noi

4.3 Pachetul PKG_STORE

Întreaga logică este încapsulată într-un singur pachet structurat, oferind o interfață clară și organizată pentru operațiunile pe baza de date.

4.3.1 Proceduri Implementate

Gestionarea Clienților:

- ADD_CLIENT: Creează simultan înregistrări în CLIENTS și CONTACT
- UPDATE_CLIENT: Actualizează informațiile din ambele tabele într-o singură operațiune
- DELETE_CLIENT: Șterge cascadă din CONTACT și CLIENTS respectând integritatea referențială

Managementul Jocurilor:

- ADD_GAME: Include validarea datei de lansare cu excepție personalizată (-20011) pentru jocurile cu lansare în viitor
- UPDATE_GAME: Permite modificarea prețurilor și stocurilor cu validări de integritate

- DELETE_GAME: Elimină jocurile din sistem cu verificarea dependențelor

Procesarea Comenzilor:

- PLACE_ORDER: Implementează logică complexă de verificare stoc, plasare comandă și actualizare inventar
- CANCEL_ORDER: Anulează comenzi cu restaurarea automată a stocului

4.3.2 Funcții Specializate

- GET_STOCK: Returnează stocul disponibil pentru un joc
- GET_CLIENT_ORDERS:
  - Returnează un SYS_REFCURSOR in care se afla informatii despre identificatorul comenzii, numele jocului comandat, cantitatea in care a fost comandat, pretul jocului la momentul comenzii si costul total al comenzii
  - Implementează JOIN între ORDERS și GAMES pentru informații complete
  - Calculează totalul comenzii folosind expresii în SELECT
  - Permite parcurgerea lazy a rezultatelor

4.3.3 Utilizarea Cursorilor:

Funcția GET_CLIENT_ORDERS demonstrează utilizarea cursorilor expliciți pentru returnarea de date complexe, oferind flexibilitate în procesarea rezultatelor.

4.3.4 Gestionarea Excepțiilor

Strategia de Gestionare a Excepțiilor:

- SAVEPOINT-uri pentru rollback în fiecare procedură
- Excepții personalizate care au coduri și mesaje specifice
- Gestionarea explicită a NO_DATA_FOUND când e cazul
- Re-ridicarea excepțiilor pentru propagarea erorilor mai departe