**BIBLIOTECĂ DIGITALĂ**

**Candidat: Maria-Alexandra Cristea**

**Coordonator științific: Ș.l.dr.ing. Mădălin-Dorin Pop**

**REZUMAT**

Această lucrare prezintă etapele dezvoltării unei baze de date pentru gestionarea unei biblioteci digitale, începând de la proiectarea conceptuală până la implementarea finală. Proiectul urmărește să ofere o soluție completă pentru evidența cărților, autorilor, genurilor literare, utilizatorilor și împrumuturilor, cu scopul de a asigura o gestionare eficientă.

În prima etapă, a fost realizată proiectarea conceptuală prin crearea unei diagrame Entitate-Relatie (ER). Aceasta include șase entități principale: Utilizatori, Împrumuturi, Carte, Detalii\_Carte, Autor și Gen\_Literar, împreună cu relațiile dintre ele de tip 1:1, 1:N și N:N. Au fost evitate anomaliile de conectare (capcanele de tip fan trap și chasm trap), iar structura bazei de date a fost optimizată prin aplicarea principiilor normalizării.

Proiectarea logică a transpus modelul conceptual în tabele relaționale, identificând cheile primare și cheile străine.

În etapa de proiectare fizică, au fost alocate tipuri de date și dimensiuni adecvate fiecărui atribut. Cheile primare și străine asigură relaționarea corectă între tabele.

Implementarea bazei de date în Oracle APEX a inclus crearea tabelelor, popularea lor cu înregistrări pentru fiecare tabelă. Au fost implementate: o vedere simpla, o vedere complexa, procedura pentru adăugarea unei carti in tabela Carte și alta pentru afișarea detaliilor unei cărți pe baza unui id specific.

# ABSTRACT

# The aim of this project is to design and implement a relational database for managing a library system, which includes functionalities for handling books, authors, users, and book loans.

# The project was carried out using Oracle Database and PL/SQL procedures to automate the data management processes.

# The first step involved creating a conceptual and logical model of the database, ensuring normalization to avoid redundancy and anomalies. The next phase included implementing the database schema with tables and relationships, followed by the development of PL/SQL procedures for data insertion, updates, and querying. Views were also created to simplify access to complex information.

# The results of the project demonstrate the effectiveness of the solution in managing library data, providing an organized and scalable structure. Future development directions include extending the system with additional features such as book reservations,

# notifications for overdue books, and generating detailed reports.1. INTRODUCERE

* 1. **SCOPUL PROIECTULUI**

Proiectul își propune să dezvolte un sistem de gestionare a unei biblioteci digitale, care să faciliteze administrarea cărților și interacțiunea utilizatorilor cu functionalitatile disponibile. Acesta oferă funcționalități precum împrumutarea și gestionarea informațiilor despre autori, genuri literare și detalii specifice fiecărei cărți.

Scopul principal al acestui proiect este de a dezvolta o bază de date pentru gestionarea eficientă a unei biblioteci digitale. Proiectul urmărește să faciliteze organizarea resurselor, să optimizeze procesele de împrumut și să asigure accesul rapid la informații relevante despre cărți, autori, utilizatori și alte entități implicate. Baza de date este proiectata să răspundă atât nevoilor administrative ale bibliotecii, cât și ale utilizatorilor.

* 1. **OBIECTIVELE PROIECTULUI**

Obiectivele proiectului sunt: crearea unei baze de date bine structurate si implementarea unui set de funcționalități de bază. Mai exact, realizarea unui model relațional care să reflecte corect relațiile dintre entități, să prevină anomaliile de actualizare, inserare și ștergere și să asigure consistența datelor si dezvoltarea unui sistem care să permită gestionarea împrumuturilor, adăugarea de noi cărți și afișarea informațiilor relevante prin utilizarea procedurilor.

* 1. **ORGANIZAREA DOCUMENTAȚIEI PROIECTULUI**

Documentația proiectului este organizată astfel:

1. Introducere – Prezintă scopul proiectului, obiectivele principale și modul de organizare a documentației.
2. Analiza stadiului actual în domeniul problemei – Oferă un rezumat al cercetărilor și exemplelor existente de baze de date relaționale similare.
3. Bazele teoretice – Explică conceptele fundamentale utilizate în proiectarea și implementarea bazei de date.
4. Soluția propusă și metodologia de proiectare/dezvoltare – Detaliază etapele de proiectare conceptuală, logică și fizică.
5. Implementare – Descrie implementarea bazei de date în Oracle Apex, incluzând scripturile pentru tabele, proceduri și funcții.
6. Utilizare și rezultate experimentale – Prezintă rezultatele obținute în urma rulării funcționalităților bazei de date.
7. Concluzii și direcții de dezvoltare – Rezumă contribuțiile proprii și evidențiază posibilele îmbunătățiri viitoare.
8. Bibliografie – Include sursele de inspirație și cercetările utilizate, în format IEEE.
9. **ANALIZA STADIULUI ACTUAL ÎN DOMENIUL PROBLEMEI**
   1. **CERCETĂRI ACTUALE ÎN DOMENIUL PROIECTĂRII ȘI IMPLEMENTĂRII BAZELOR DE DATE RELAȚIONALE**

În articolul „Handling Big Data in Relational Database Management Systems”, publicat pe site-ul ScienceDirect, se discută despre diferite strategii de stocare a datelor, împărțite în trei categorii: structurate, nestructurate și semi-structurate. Stocarea datelor structurate se realizează eficient în sistemele de gestionare a bazelor de date relaționale (RDBMS), care folosesc un format tabular cu tabele și relații predefinite. Exemple comune de utilizare includ tranzacțiile de vânzări și sistemele de rezervare. Stocarea datelor nestructurate se referă la fișiere video și imagini, care necesită securitate sporită, de exemplu în domeniul medical. Există trei strategii pentru stocarea video: în sistemul de fișiere, în baze de date (BLOB) și o soluție hibridă numită "data links". Fiecare are avantaje și dezavantaje, inclusiv probleme de eficiență și gestionare a metadatelor. Stocarea datelor semi-structurate folosește modelul de date JSON, care permite crearea unui scheme flexibile în RDBMS pentru a gestiona relațiile și atributele multivalente. JSON este utilizat pentru serializarea și transmiterea datelor și este implementat în diverse baze de date relaționale, precum MySQL și PostgreSQL, cu performanțe mai bune decât soluțiile NoSQL pentru volume mici de date.

In articolul “Relational database schema design for uncertain data” publicat pe site-ul ScienceDirect, designul schemei bazei de date relaționale pentru date incerte presupune atribuirea de grade de posibilitate tuplurilor și certitudine dependențelor funcționale, permițând controlul integrității datelor, eficienței interogărilor și actualizărilor. Dependințele funcționale posibile (pFD) sunt utilizate pentru a modela incertitudinea în date, stabilind un cadru teoretic care permite crearea unor scheme normalizate, echilibrând integritatea și pierderea informațiilor.

Lucrarea propune un cadru bazat pe pFD, care definește concepte precum redundanța valorilor și anomaliile de actualizare și introduce forme normale pentru a rezolva aceste probleme și a asigura validarea eficientă a schemelor. Modelul permite gestionarea compromisurilor între integritatea datelor, pierderea acestora și eficiența interogărilor și actualizărilor prin controlul gradului de certitudine. Cadrul oferă un mecanism de justificare a schemelor ne-normalizate în funcție de nivelul dorit de integritate și pierdere a datelor.

Lucrarea abordează problema implicației pentru pFD și prezintă un algoritm eficient pentru rezolvarea acesteia, adaptat la diferitele grade de incertitudine. De asemenea, sunt introduse forme normale semantice pentru p-relații și sunt stabilite corespondențe cu formele normale clasice din bazele de date relaționale. Normalizarea bazelor de date posibile presupune normalizarea clasică în funcție de cerințele aplicației, iar decompozițiile BCNF elimină redundanțele și păstrează integritatea dependențelor funcționale.

Teoria posibilităților oferă o abordare calitativă pentru a gestiona incertitudinea în date, permițând proiectarea de scheme relaționale care reduc redundanța și elimină anomaliile de actualizare. Aceste metode optimizează performanța bazelor de date prin reducerea erorilor și îmbunătățirea eficienței proceselor de interogare și actualizare.

* 1. **BAZE DE DATE EXISTENTE SIMILARE CELEI PROPUSE**

Proiectul *Digital Victorian Periodical Poetry (DVPP)* explorează utilizarea bazelor de date relaționale, în special MySQL, pentru gestionarea metadatelor despre poeziile publicate în reviste britanice din secolul al XIX-lea. Este o inițiativă interdisciplinară care îmbină cercetarea literară cu instrumentele digitale, scopul fiind de a organiza, analiza și prezenta informații detaliate despre poeziile și revistele respective.

Structura bazei de date MySQL în DVPP:

1. Tabelul principal: "Poems" (Poezii): fiecare înregistrare corespunde unei poezii publicate într-o anumită revistă, la o anumită dată.

Atribute:

* + - ID unic al poeziei.
    - Titlu.
    - Dată publicare.
    - ID revistă (legătură cu tabelul "Organs").

1. Tabelul "Organs" (Reviste): păstrează informații despre revistele în care au apărut poeziile.

Atribute:

* + - ID unic revistă.
    - Nume revistă.
    - Informații adiționale despre publicație.

1. Tabelul "People" (Persoane): conține detalii despre persoanele implicate în crearea poeziilor: autori, traducători, ilustratori.

Atribute:

* + - ID unic persoană.
    - Nume.
    - Rol (autor, traducător, etc.).

1. Relații între tabele:

* poate avea mai mulți autori, iar un autor poate fi asociat cu mai multe poezii
* Poeziile sunt legate de reviste printr-un ID comun.
* Persoanele sunt conectate la poezii printr-un tabel intermediar, care specifică rolurile lor.

# BAZELE TEORETICE

În realizarea acestui proiect, mai multe concepte fundamentale din domeniul bazelor de date relaționale și al programării SQL/PLSQL au fost aplicate.

* 1. **PROIECTAREA CONCEPTUALĂ**

Baza de date reprezinta un set de date interconectate și organizate în scopul prelucrării rapide de către mai multe persoane.

Proiectarea conceptuală a bazei de date este o etapă fundamentală în dezvoltarea unui sistem informatic, având rolul de a reprezenta informațiile într-o manieră abstractă, fără a se preocupa încă de detaliile tehnice ale implementării. Aceasta presupune crearea unui model conceptual care să descrie obiectele de interes, relațiile dintre ele și caracteristicile esențiale, fără a face referire la detalii de implementare specifice. Un aspect important al proiectării conceptuale este utilizarea modelului Entitate-Relatie (ER), care ajută la structurarea și organizarea logică a datelor.

Modelele Entitate-Relatie sunt deosebit de utile pentru că facilitează vizualizarea interacțiunilor dintre diverse entități și relațiile dintre ele, ajutând astfel la identificarea unui design eficient și coerent al bazei de date.

O entitate reprezintă un obiect, persoană, concept sau orice alt element distinct care poate fi identificat, modelat ca obiect independent și care este de interes pentru sistemul de date.

Atributele sunt caracteristici sau proprietăți care descriu o entitate sau o relație.

O relație reprezintă asocierea dintre două sau mai multe entități. Tipuri de relatii:

* Unu-la-unu (1:1): O entitate este asociată cu exact o altă entitate.
* Unu-la-mulți (1:N): O entitate este asociată cu mai multe entități.
* Mulți-la-mulți (M:N): Mai multe entități dintr-un set sunt asociate cu mai multe entități din alt set.

Exista două categorii de capcane de conectare:

▪ Capcane în "evantai" (fan traps): aceiași entitate este implicată în mai multe relații de tip 1 : N în care căile dintre entități devin ambigue;

▪ Capcanele de întrerupere (chasm traps): omiterea reprezentării unor relații în modelul ER.

Pentru a preveni astfel de capcane, proiectarea conceptuală trebuie să fie realizată cu atenție, asigurându-se că toate entitățile și relațiile sunt bine definite și că nu există ambiguități în modelul ER. În plus, schema conceptuală obținută este un ghid esențial care va fi utilizat în etapele ulterioare ale proiectării bazei de date, inclusiv în faza de implementare fizică a acesteia.

Schema conceptuală reprezinta totalitatea obiectelor obținute în etapa de proiectare conceptuală (entități, atribute, relații).

* 1. **PROIECTAREA LOGICĂ**

Proiectarea logică reprezintă o etapă esențială în crearea unei baze de date, având scopul de a organiza datele într-o structură eficientă și de a stabili relațiile între acestea. În această fază, se pune accent pe crearea unei viziuni clare asupra modului în care datele vor fi interconectate, fără a ține cont de implementarea fizică, care va urma în etapa de proiectare fizică. Astfel, structura logică a bazei de date se construiește pe baza unui model abstract care reflectă cerințele funcționale ale sistemului.

Cheia primară este un atribut sau un set de atribute care identifică în mod unic fiecare înregistrare (instanță) dintr-o tabelă.

O cheie străină este un atribut sau un set de atribute dintr-o tabelă care face referire la cheia primară a altei tabele. Ea definește relația dintre două tabele.

Normalizarea reprezintă o practică importantă în proiectarea logică a bazei de date, având ca scop reducerea redundanței datelor și îmbunătățirea eficienței bazei de date. Prin normalizare, datele sunt organizate în tabele separate, astfel încât fiecare tabel să conțină doar informațiile esențiale și să minimizeze riscurile de inconsistență. Etapele de normalizare sunt esențiale pentru obținerea unei baze de date eficiente și corecte, începând cu prima formă normală (1NF) și continuând cu formele 2NF, 3NF și 4NF, fiecare având scopul de a elimina diverse tipuri de redundanță și dependențe inutile.

Cele patru forme normale sunt:

▪ forma normală 1 (1NF): fiecare câmp dintr-o înregistrare va conține o singura valoare;

▪ forma normală 2 (2NF): trebuie să existe dependenţă funcţională completă faţă de cheia primară;

▪ forma normală 3 (3NF): să nu existe câmpuri, care nu fac parte din cheia primară şi care să determine alt câmp (sa nu existe dependente tranzitive);

▪ forma normală 4 (4NF): să nu existe dependenţe multivalorice în baza de date.

De asemenea, procesul de descompunere a tabelelor joacă un rol important în proiectarea logică, deoarece permite împărțirea unui tabel complex într-o serie de tabele mai simple, fiecare având o structură clară și bine definită. Această descompunere ajută nu doar la îmbunătățirea performanței bazei de date, dar și la menținerea unei structuri coerente și logice, care este mai ușor de gestionat și întreținut.

În ceea ce privește relațiile dintre entități, acestea pot fi reprezentate prin tabele asociative atunci când există o relație de tip mulți-la-mulți (M:N), ceea ce ajută la menținerea unei structuri corecte a bazei de date, evitând redundanțele. Cheile străine sunt adesea folosite pentru a crea aceste relații, asigurându-se astfel că legăturile între tabele sunt bine definite și respectă regulile de integritate referențială.

* 1. **PROIECTAREA FIZICĂ**

Modelul fizic reprezintă implementarea bazei de date. Modelul fizic al unei baze de date reprezintă nivelul cel mai detaliat al proiectării bazei de date, care se concentrează pe modul în care datele sunt stocate, organizate și accesate.

Pasi esențiali în trecerea de la modelul logic la cel fizic:

▪ obținerea modelelor tabelelor relaționale din modelul logic global;

▪ identificarea constrângerilor impuse datelor (opționalitate, PK, FK);

▪ identificarea tipului de date și a dimensiunii de stocare pentru fiecare atribut;

▪ identificarea nivelului de securitate necesar bazei de date dezvoltate;

▪ introducerea de denumiri prescurtate pentru atribute cu scopul de a simplifica implementarea modelului cu condiția includerii unei descrieri pentru fiecare denumire prescurtată.

Un pas esențial în trecerea de la modelul logic la modelul fizic este realizarea unei transpuneri corecte a tabelelor din modelul logic, având în vedere toate relațiile și constrângerile impuse de modelul anterior. Aceasta include identificarea și definirea tipurilor de date specifice pentru fiecare atribut, precum și stabilirea dimensiunilor de stocare adecvate pentru a asigura performanță optimă în manipularea datelor.

De asemenea, în proiectarea fizică a bazei de date, este importantă definirea detaliată a constrângerilor de integritate, inclusiv cheile primare și externe (PK, FK), precum și stabilirea unor constrângeri suplimentare legate de opționalitatea datelor. Aceste constrângeri contribuie la asigurarea corectitudinii și consistenței datelor pe măsură ce acestea sunt gestionate și stocate.

Un alt aspect important în această fază este definirea măsurilor de securitate necesare pentru protejarea bazei de date. Se analizează nivelurile de acces, drepturile utilizatorilor și mecanismele de autentificare, astfel încât baza de date să fie protejată împotriva accesului neautorizat și a atacurilor externe.

În plus, pentru a ușura implementarea și a spori eficiența managementului bazei de date, se recomandă utilizarea denumirilor prescurtate pentru atributele și tabelele, cu condiția ca acestea să fie descrise în mod corespunzător în documentația bazei de date. Aceste denumiri ajută la simplificarea codului și a proceselor de administrare, fără a compromite claritatea și înțelegerea structurii bazei de date.

* 1. **INSTRUCȚIUNI DDL**

Instrucțiunile DDL (Data Definition Language) sunt utilizate pentru crearea, modificarea și eliminarea obiectelor din schemă, cum ar fi tabelele, indexurile și vederile.

Vederile (views) sunt reprezentări logice sub formă tabelară, rezultate din interogarea uneia sau mai multor tabele sau vederi existente. Acestea pot fi de două tipuri: vederi simple, care furnizează date dintr-o singură tabelă de bază, nu conțin funcții sau grupuri de date și permit execuția de operații DML (INSERT, UPDATE și DELETE), și vederi complexe, care furnizează date din mai multe tabele de bază, pot conține funcții sau grupuri de date, dar nu permit întotdeauna operații DML.

Constrângerile (constraints) sunt reguli specifice care asigură integritatea bazei de date, prevenind introducerea datelor invalide. Acestea pot fi specificate la crearea unei tabele prin instrucțiunea CREATE TABLE sau adăugate ulterior cu ajutorul instrucțiunii ALTER TABLE. Constrângerile pot fi aplicabile unor anumite atribute ale unei tabele sau întregii tabele.

Pe lângă DDL, limbajul DML (Data Manipulation Language) include comenzi principale precum INSERT, UPDATE, DELETE, și SELECT, precum și subinterogări utilizând operatori precum UNION, INTERSECT și MINUS.

De asemenea, limbajul PL/SQL este folosit pentru dezvoltarea și administrarea bazelor de date, permițând crearea de blocuri anonime, proceduri și funcții stocate, declanșatori (triggers) și pachete care îmbunătățesc funcționalitatea și automatizarea operațiunilor în cadrul bazelor de date.

1. **SOLUȚIA PROPUSĂ ȘI METODOLOGIA DE PROIECTARE/DEZVOLTARE**

Proiectul realizat are scopul de a crea o bază de date relațională pentru gestionarea unei biblioteci digitale, permițând stocarea și manipularea informațiilor despre utilizatori, cărți, autori, genuri literare și împrumuturi. Metodologia de proiectare a implicat trei etape principale: proiectarea conceptuală, proiectarea logică și proiectarea fizică. Fiecare etapă a fost abordată, iar detaliile soluției propuse sunt prezentate în cele ce urmează.

* 1. **PROIECTAREA CONCEPTUALĂ**

În această etapă, am construit diagrama Entitate-Relatie (ER), care descrie structura conceptuală a bazei de date, inclusiv entitățile principale și relațiile dintre ele. În cadrul proiectului, au fost identificate următoarele entități: Utilizator, Carte, Autor, Gen Literar, Împrumut și Detalii Carte. Fiecare entitate a fost definită, după cum urmează:

* Utilizator: id\_utilizator (identificator unic), nume\_utilizator, email, data\_inregistrare, oras, cod\_postal, nr\_telefon.
* Carte: id\_carte (identificator unic), titlu\_carte, data\_publicare, limba.
* Autor: id\_autor (identificator unic), nume\_autor, nationalitate, data\_nastere\_autor.
* Gen Literar: id\_gen (identificator unic), nume\_gen, descriere\_gen.
* Împrumut: id\_imprumut (identificator unic), data\_imprumut, data\_retur, status.
* Detalii Carte: id\_detalii\_carte, nr\_pagini, descriere.

Relațiile dintre entități au fost definite astfel:

* 1:1 între Carte și Detalii Carte: fiecare carte are un set unic de detalii.
* 1:N între Autor și Carte: un autor poate scrie mai multe cărți, dar fiecare carte are un singur autor.
* 1:N intre Gen Literar si Carte: un gen poate include mai multe carti.
* 1:N Imprumut si Utilizator: un utilizator poate avea mai multe imprumuturi, fiecare imprumut e asociat cu un singur utilizator.
* N:N între Carte și Împrumut: o carte poate fi împrumutată de mai mulți utilizatori, iar un împrumut poate conține mai multe cărți. Această relație a fost implementată prin intermediul unei tabele de legătură, Carte\_Imprumut.

Capcanele de conectare:

* Fan traps: S-a analizat fiecare relație pentru a preveni conexiuni redundante. De exemplu, relația dintre Gen Literar și Carte este directă, fără alte intermedieri care ar putea complica interogările.
* Chasm traps: Am verificat că toate relațiile esențiale sunt acoperite și nu există lipsuri între tabele, asigurându-se că fiecare entitate este corect conectată.

În urma proiectarii conceptuale, am obtinut urmatoarea diagrama entitate-relatie:

A diagram of a flowchart

Description automatically generated

Figura 1 – Diagrama Entitate-Relație

* 1. **PROIECTAREA LOGICĂ**

În această etapă a proiectării bazei de date, diagrama ER a fost transformată într-un model relațional, detaliind structura bazei de date în termeni tehnici și asigurând normalizarea tabelelor pentru a elimina redundanțele și a îmbunătăți integritatea datelor.

Primul pas important a fost identificarea cheilor primare și străine. Fiecare tabelă a fost asociată cu o cheie primară (PK) – un atribut sau un set de atribute care identifică unic fiecare înregistrare din tabelă. De exemplu, în tabela „Utilizator”, atributul „id\_utilizator” este utilizat ca cheie primară, iar în tabela „Împrumut”, „id\_imprumut” joacă rolul de cheie primară. Pe lângă acestea, au fost definite chei străine (FK), care asigură relațiile între tabele. Astfel, în tabela „Carte”, „id\_autor” este o cheie străină care face legătura cu tabela „Autor”, iar „id\_gen” din tabela „Carte” se referă la genurile literare din tabela „Gen\_Literar”.

A computer screen with pink and white text

Description automatically generated

Figura 2 – Modelul logic inițial

Un alt aspect important a fost aplicarea procesului de normalizare a tabelelor pentru a elimina redundanțele și a asigura integritatea datelor.

Astfel, am aplicat 1NF pentru a elimina valorile multiple dintr-un camp, astfel încât fiecare câmp să conțină o singură valoare, conform cerinței 1NF.

Înainte de normalizare, tabela „Utilizator” ar fi putut conține un câmp „nr\_telefon” în care ar fi fost stocate mai multe numere de telefon pentru un singur utilizator. Acest lucru încalcă principiile 1NF, care stipulează că fiecare câmp al unei tabele trebuie să conțină o singură valoare atomică.

Pentru a adera la 1NF, am transformat câmpul „nr\_telefon” într-o formă care să conțină doar un singur număr de telefon/înregistrare. Astfel, dacă un utilizator are mai multe numere de telefon, am creat o relație separată, sub forma unei tabele suplimentare, care conține câte un rând pentru fiecare număr de telefon asociat unui utilizator. Astfel, am creat o tabela Numar\_Telefon care are cheia primara id\_telefon, cheia straina id\_utilizator si atributul nr\_telefon. Tabela Utilizator si Numar\_Telefon sunt legate prin cheia straina id\_utilizator.

A screenshot of a computer

Description automatically generated

Figura 3 – Modelul logic obținut în urma normalizarii 1NF

Apoi, am aplicat 3NF pe tabela Utilizator, pentru ca am identificat că există o dependență tranzitivă între atributele cod postal și oraș. Conform 3NF, trebuie să eliminăm dependențele tranzitive, adică atunci când un atribut non-cheie depinde de un alt atribut non-cheie.

In cazul de fata, un cod postal specific poate corespunde unui singur oraș. Astfel, orașul poate fi derivat din codul poștal. Am creat o noua tabela, Locatie, care contine atributele oras si cod postal, care se aflau initial in tabela Utilizator, si cheia primara id\_locatie.

După crearea tabelei Locatie, am eliminat cod postal și oraș din tabela Utilizator și am adăugat id\_locatie ca si cheia străină, care face legătura între Utilizator și tabela Locatie. Astfel, informațiile despre oraș și codul poștal sunt acum gestionate într-o manieră separată și nu mai există redundanțe sau dependențe tranzitive.

A screenshot of a computer

Description automatically generated

Figura 4 – Modelul logic obținut în urma normalizarii 3NF

De asemenea, s-au luat măsuri pentru a preveni anomaliile în cadrul operațiilor de inserare, actualizare și ștergere. În ceea ce privește anomaliile de inserare, am asigurat că toate tabelele permit introducerea de date fără a necesita informații redundante. În cazul anomaliilor de actualizare, datele sunt stocate într-un singur loc, de exemplu, actualizarea unui gen literar se face exclusiv în tabela „Gen\_Literar”. În ceea ce privește anomaliile de ștergere, am asigurat că ștergerea unei cărți nu duce la pierderea informațiilor legate de autorul acesteia.

* 1. **PROIECTAREA FIZICĂ**

În cadrul etapei de proiectare fizică a bazei de date, s-a realizat implementarea modelului relațional, detaliind aspectele fundamentale ale structurii bazei de date, inclusiv constrângerile impuse datelor, tipurile de date și denumirile câmpurilor.

Pentru a asigura integritatea și corectitudinea datelor, au fost impuse mai multe constrângeri. Printre acestea se numără PRIMARY KEY, care garantează unicitatea înregistrărilor din tabele, și FOREIGN KEY, care definește relațiile între tabele, menținând integritatea referențială. De asemenea, au fost utilizate constrângeri de tip NOT NULL, pentru a marca câmpurile obligatorii ce nu pot rămâne goale, și UNIQUE pentru câmpuri care nu permit duplicate, cum ar fi adresa de email din tabela „Utilizator”.

Pentru tipurile de date, fiecare atribut a fost asociat cu tipul de date cel mai potrivit, pentru a optimiza performanța și a asigura o stocare eficientă a datelor. Astfel, am folosit VARCHAR2 pentru atributele de tip text scurt, precum numele utilizatorului, INT pentru identificatori unici și atribute numerice, cum ar fi ID-ul cărții, DATE pentru date calendaristice, precum data împrumutului, și CLOB pentru stocarea textelor extinse, de exemplu, pentru descrierea genurilor literare.

În ceea ce privește denumirile câmpurilor, acestea au fost alese astfel încât să fie intuitive și clare, iar fiecare atribut a fost însoțit de o descriere precisă. De exemplu, atributul status din tabela „Împrumut” indică dacă împrumutul este activ sau finalizat, iar descriere\_gen din tabela „Gen\_Literar” oferă informații suplimentare despre genul literar asociat fiecărei cărți. Aceste denumiri și descrieri ajută la o mai bună înțelegere a funcționalității fiecărei tabele și atribut din cadrul bazei de date.

A black screen with white text

Description automatically generated

Tabelul 1 – Model Relațional Tabela Carte

A screen shot of a computer

Description automatically generated

Tabelul 2 – Model Relațional Tabela Autor

A black screen with white text

Description automatically generated

Tabelul 3 – Model Relațional Tabela Gen literar

A black screen with white text

Description automatically generated

Tabelul 4 – Model Relațional Tabela Detalii carte

A black screen with white text

Description automatically generated

Tabelul 5 – Model Relațional Tabela Împrumut

A screen shot of a computer

Description automatically generated

Tabelul 6 – Model Relațional Tabela Utilizator

A black screen with white text

Description automatically generated

Tabelul 7 – Model Relațional Tabela Locație

A computer screen with a black background

Description automatically generated

Tabelul 8 – Model Relațional Tabela Carte-Împrumut

A black screen with white text

Description automatically generated

Tabelul 9 – Model Relațional Tabela Număr\_Telefon

**5. IMPLEMENTARE**

Etapa de implementare a constat în realizarea modelului relațional, configurarea bazei de date în mediul Oracle Apex și scrierea de cod pentru gestionarea operațiilor cerute. Această secțiune detaliază procesul de creare de tabele, popularea acestora cu date, crearea vederilor și realizarea de proceduri PL/SQL.

**5.1. CREAREA TABELELOR**

Modelul relațional definit în etapa de proiectare fizică a fost implementat folosind comenzi SQL CREATE TABLE. Fiecare tabelă a fost creată cu respectarea constrângerilor identificate anterior.

A screenshot of a computer

Description automatically generated

Figura 5 – Crearea tabelei Locație

A screenshot of a computer

Description automatically generated

Figura 6 – Crearea tabelei Împrumut

A screenshot of a computer

Description automatically generated

Figura 7 - Crearea tabelei Gen\_Literar

A screenshot of a computer

Description automatically generated

Figura 8 – Crearea tabelei Detalii\_Carte

A screenshot of a computer program

Description automatically generated

Figura 9 – Crearea tabelei Carte\_Împrumut

A screenshot of a computer program

Description automatically generated

Figura 10 – Crearea tabelei Carte

A screenshot of a computer program

Description automatically generated

Figura 11 – Crearea tabelei Autor

A screenshot of a computer

Description automatically generated

Figura 12 – Crearea tabelei Numar\_Telefon

CREATE TABLE Utilizator(

id\_utilizator NUMBER PRIMARY KEY,

nume\_utilizator VARCHAR(50) NOT NULL,

email VARCHAR(30) NOT NULL UNIQUE,

data\_inregistrare DATE NOT NULL,

id\_locatie INT,

FOREIGN KEY (id\_locatie) REFERENCES Locatie(id\_locatie)

);

**5.2. POPULAREA TABELELOR CU DATE**

Înregistrarea datelor într-o tabelă se referă la procesul de adăugare a unui nou set de informații într-o tabelă dintr-o bază de date. Fiecare înregistrare este un rând din tabelă, iar fiecare câmp (coloană) din acel rând conține o valoare specifică care reprezintă o unitate de informație.

Pentru a valida funcționarea modelului relațional, au fost introduse cel puțin 5 înregistrări în fiecare tabelă folosind comenzi INSERT INTO.

A screenshot of a computer

Description automatically generated

Figura 13 – Inserarea unei înregistrări în tabela Carte\_Împrumut

A screenshot of a computer

Description automatically generated

Figura 14 – Inserarea unei înregistrări în tabela Gen\_Literar

A screenshot of a computer

Description automatically generated

Figura 15 – Inserarea unei înregistrări în tabela Împrumut

A screenshot of a computer

Description automatically generated

Figura 16 – Inserarea unei înregistrări în tabela Carte

A screenshot of a computer

Description automatically generated

Figura 17 – Inserarea unei înregistrări în tabela Locație

A screenshot of a computer

Description automatically generated

Figura 18 – Inserarea unei înregistrări în tabela Autor

A screenshot of a computer

Description automatically generated

Figura 19 – Inserarea unei înregistrări în tabela Utilizator

A screenshot of a computer

Description automatically generated

Figura 20 – Inserarea unei înregistrări în tabela Numar\_Telefon

**5.3. PROCEDURĂ / FUNCTIE PL/SQL PENTRU ADĂUGAREA UNEI NOI ÎNREGISTRĂRI**

Procedura PL/SQL, denumită AdaugaCarte, adaugă o nouă înregistrare în tabela Carte. Procedura primește trei parametri de intrare: p\_titlu\_carte, p\_data\_publicare și p\_limba. În cadrul procedurii se determină un id\_carte unic, folosind valoarea maximă existentă în tabela Carte și adăugând 1 la aceasta; se adaugă o nouă înregistrare în tabela Carte cu datele primite și cu id\_carte generat; se face un commit pentru a salva modificările în baza de date; dacă adăugarea are succes, se afișează mesajul „Cartea a fost adăugată cu succes!”; in caz de eroare, se face un rollback pentru a anula modificările și se afișează mesajul de eroare.

A screenshot of a computer program

Description automatically generated

Figura 21 – Procedura AdaugaCarte

**5.4. PROCEDURĂ / FUNCTIE PL/SQL PENTRU AFIȘAREA DATELOR DIN DOUĂ SAU MAI MULTE TABELE**

Această procedură PL/SQL, denumită afisare\_detalii\_carte, este destinată pentru a afișa detaliile complete ale unei cărți, utilizând un parametru de intrare p\_id\_carte, care reprezintă identificatorul cărții pentru care dorim să obținem informațiile.

Procedura afisare\_detalii\_carte permite afișarea completă a detaliilor unei cărți din baza de date, inclusiv informații despre autor, gen literar și detalii suplimentare, folosind un singur parametru p\_id\_carte.

Procedura primește un parametru de tip VARCHAR2 denumit p\_id\_carte, care reprezintă ID-ul cărții pentru care se vor afișa detaliile. În cadrul procedurii, se execută o interogare JOIN între patru tabele (Carte, Autor, Gen\_Literar, și Detalii\_Carte), pentru a aduna informațiile complete despre carte. Pentru fiecare înregistrare obținută în urma interogării se afișează detaliile cărții folosind DBMS\_OUTPUT.PUT\_LINE. Dacă nu există nicio înregistrare în tabelă cu id\_carte corespunzător valorii primite, se va afișa un mesaj care indică faptul că nu s-au găsit date pentru acest ID. Dacă apare orice altă eroare, se va captura și afișa mesajul de eroare asociat.

A screenshot of a computer

Description automatically generated

Figura 22 – Procedură de afișare detalii a)

A screenshot of a computer program

Description automatically generated

Figura 23 – Procedură de afișare detalii b)

**5.5. CREAREA UNEI VEDERI SIMPLE SI UNEI VEDERI COMPLEXE**

Vederea Vedere\_Titluri\_Carti este o vedere simpla. Scopul acestei vederi este de a oferi o reprezentare simplă a informațiilor de bază despre cărți, referindu-se la trei coloane esențiale: id\_carte, titlu\_carte, limba. Această vedere este generată printr-o instrucțiune SQL simplă, care selectează cele trei coloane menționate din tabelul principal al bazei de date, denumit Carte.

**A screenshot of a computer

Description automatically generated**

Figura 24 – Vedere simpla

Vedrea Vedere\_Carti este o vedere complexa. Vedere\_Carti combină informații din două tabele ale bazei de date: Carte și Detalii\_Carte. Scopul acestei vederi este de a furniza o reprezentare mai detaliată a datelor, incluzând atât titlurile cărților, cât și informații suplimentare.

Vederea este construită printr-o interogare SQL cu JOIN, care asociază datele din tabelul Carte cu cele din tabelul Detalii\_Carte, pe baza coloanei comune id\_carte.

Instrucțiunea JOIN permite combinarea informațiilor din cele două tabele, astfel încât să se obțină un set de date unificat care conține atât titlurile cărților, cât și detaliile despre numărul de pagini.

Această vedere selectează identificatorul unic al cărții, pentru a asigura că fiecare carte poate fi diferențiată , afișează titlul cărții, oferind o informație de interes, include numărul de pagini al fiecărei cărți.

A screenshot of a computer

Description automatically generated

Figura 25 – Vedere complexă

**6. UTILIZARE, REZULTATE EXPERIMENTALE**

În această secțiune, vor fi prezentate rezultatele obținute în urma rulării implementării cerințelor de la punctul 4, care se referă la crearea tabelelor si popularea acestora, la cele doua vederi create si la manipularea datelor în baza de date, prin proceduri PL/SQL.

**6.1. CONȚINUTUL TABELELOR**

A screenshot of a computer

Description automatically generated

Figura 26 – Conținutul tabelei Carte

A screenshot of a computer

Description automatically generated

Figura 27 – Conținutul tabelei Carte\_Împrumut

A screenshot of a computer

Description automatically generated

Figura 28 – Conținutul tabelei Detalii\_Carte

A screenshot of a computer

Description automatically generated

Figura 29 – Conținutul tabelei Gen\_Literar

A screenshot of a computer

Description automatically generated

Figura 30 – Conținutul tabelei Împrumut

A screenshot of a computer

Description automatically generated

Figura 31 – Conținutul tabelei Locație

A screenshot of a computer

Description automatically generated

Figura 32 – Conținutul tabelei Utilizator

A screenshot of a computer

Description automatically generated

Figura 33 – Conținutul tabelei Autor

**6.2. VEDRE SIMPLA SI COMPLEXA**

A screenshot of a computer

Description automatically generated

Figura 34 – Vedere complexă

A screenshot of a computer

Description automatically generated

Figura 35 – Vedere simplă

**6.3. PROCEDURA DE ADAUGARE A UNEI INREGISTRARI SI PROCEDURA PENTRU AFISAREA DATELOR DIN DOUA SAU MAI MULTE TABELE**

A screen shot of a computer

Description automatically generated

Figura 36 – Rezultatul procedurii de adăugare a unei înregistrări

A black rectangular object with a black stripe

Description automatically generated

Figura 37 – Rezultatul procedurii de afișare a detaliilor unei cărți

**7. CONCLUZII ȘI DIRECȚII DE DEZVOLTARE**

**7.1. CONCLUZII**

Proiectul a avut ca scop proiectarea și implementarea unei baze de date relaționale destinate gestionării unei biblioteci. Acesta include funcționalități esențiale pentru administrarea cărților, autorilor, utilizatorilor și împrumuturilor. Contribuțiile principale în cadrul acestui proiect au fost următoarele:

* Proiectarea conceptuală și logică: Am definit un model de date care asigură eficiență și scalabilitate, având la bază principiile normalizării pentru a preveni redundanțele și anomaliile.
* Implementarea în Oracle Apex: Am realizat efectiv structura bazei de date prin crearea tabelelor, popularea acestora cu date și scrierea de proceduri PL/SQL care permit gestionarea eficientă a datelor.
* Crearea de vederi: Am simplificat accesul la informații complexe prin interogări predefinite, facilitând astfel analiza datelor din mai multe tabele relaționate.
* Automatizarea proceselor: Am implementat funcții care asigură consistența datelor și reduc posibilitatea erorilor umane în gestionarea acestora. Procedurile PL/SQL reduc riscurile asociate erorilor manuale și asigură o gestionare corectă a datelor.

Avantajele soluției propuse:

* Eficiență și structură bine organizată: Modelul relațional respectă standardele de normalizare până la a patra formă normală (4NF), evitând redundanțele și menținând integritatea datelor.
* Flexibilitate: Soluția permite adăugarea ușoară a unor noi funcționalități, cum ar fi gestionarea achizițiilor de cărți sau implementarea unui sistem de penalizări pentru întârzierea returnării acestora.
* Interogări complexe simplificate: Vederile create facilitează accesul rapid și eficient la informațiile necesare din tabelele relaționate, oferind o interogare ușoară a datelor complexe.
* Automatizare și validare: Procedurile PL/SQL reduc riscurile asociate erorilor manuale și asigură o gestionare corectă a datelor.

Dezavantajele solutiei propuse:

* Dependența de Oracle Apex: Soluția este creată pentru a funcționa în Oracle Apex, ceea ce face mai greu să o folosești pe alte platforme sau medii.
* Scalabilitate limitată: Dacă volumul de date crește foarte mult, ar putea fi necesare modificări suplimentare pentru a asigura performanța optimă a bazei de date.

**7.2. DIRECȚII DE DEZVOLTARE ULTERIOARĂ**

* Sugerarea unor cărți pe baza unor statistici (în funcție de cărțile împrumutate anterior, de varstă, de țara de proveniență, de genul literar preferat).
* Implementarea unei interfețe grafice: Ar fi utilă integrarea unei aplicații web sau mobile care să permită utilizatorilor să gestioneze împrumuturile și cărțile.
* Implementarea unei funcționalități de reminder pentru returnarea unei cărți.
* Implementarea unei funționalități pentru rezervarea cărților.

# BIBLIOGRAFIE

K. ElDahshan , E. Selim , A. Ismail Ebada,M. Abouhawwash, Y. Nam, G. Behery, *Handling Big Data in Relational Database Management Systems*,  vol. 72, pp 5149-5164, 2022.

S. Link, H. Prade, *Relational database schema design for uncertain data,* vol.84, pp 88-110, 2019*.*

M. Pop, Curs Proiectarea Bazelor de Date, Universitatea Politehnica Timișoara

F. Radulescu, Curs Baze de date, Universitatea Politehnica București,

<http://bdfr.cs.pub.ro/BD1-slides.pdf>

M. Udeac, Curs Baze de date, Colegiul Național „Petru Rareș” Suceava, <https://www.ududec.com/wp-content/uploads/2023/09/A.-Baze-de-Date.-Modulul-I-Proiectarea-bazelor-de-date.pdf>

L. Scripcariu, Curs Proiectarea Bazelor de Date, Universitatea Tehnică „Gheorghe Asachi” Iași, <http://telecom.etc.tuiasi.ro/telecom/staff/lscripca/BD1.pdf>

\*\*\* <https://www.pbinfo.ro/articole/2278/baze-de-date> accesare decembrie 2024.

\*\*\* <https://edu.info.uaic.ro/baze-de-date/ro/index.html> accesare decembrie 2024.

\*\*\* <https://dvpp.uvic.ca/womenPoets.html> accesare decembrie 2024.