Centrul de Excelență în Informatică și Tehnologii Informaționale

Catedra Informatică I

Lucrare de studiu individual Nr.1

Disciplina: Sisteme de gestiune a bazelor de date

Tema: **Baza de date privind activitatea unei companii**

Realizat: Bruma Alexandrina

Grupa: P-2331

Verificat: Covali Eugenia

Nota _____

Chișinău 2025

Cuprins

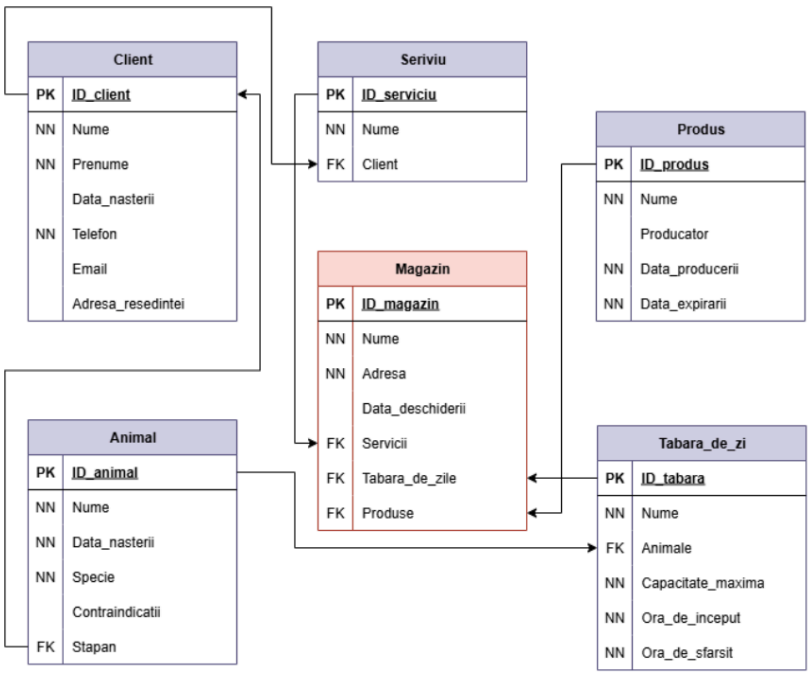
1. Introducere

În ziua de astăzi, bazele de date joacă un rol esențial în viața noastră. Ca rezultat faptului că lumea se digitalizează din ce în ce mai mult, informațiile sunt păstrate, la fel, în mod digital, prin intermediul bazelor de date, astfel, gestionarea datelor a devenit mult mai ușoară. Eu am realizat baza de date a unui magazin pentru animale, unde nu sunt prezente doar produse pentru îngrijirea lor, dar și alte servicii, precum frizerie, veterinar și chiar o tabără de zi, unde stăpânii care nu au posibilitatea de a-și îngriji animalele pe parcursul zilei, le pot lăsa pe mâini bune până se eliberează. Prin urmare, datorită bazelor de date modern, gestionarea unui asemenea magazine, cu multiple filiale și diverse servicii, devine mult mai comodă și ușoară.

2. Elementele de proiectare a bazei de date:

2.1. Descrierea modelului (diagrama Entitate-Relație):

Diagrama ER (Entity Relationship Diagram) - diagramă care afișează relația dintre seturile de entitățile, ele fiind obiectele principale care se descriu într-o bază de date. ERD-urile Sunt adesea folosite pentru a proiecta sau depana erori în baze de date relaționale. În această lucrare vom explora aspectele fundamentale ale proiectării unei baze de date, concentrându-ne pe modelul **Entitate-Relație**, pe modelul de date relațional, precum și pe modul în care acestea sunt implementate în practică.



2.2. Descrierea modelelor de date. Modelul de date relațional:

Modelul de date relațional a fost creat de cercetătorul E.F. Codd de la IBM, care în 1970 a publicat lucrarea intitulată „Un model relațional de date pentru mari colecții de date partajate”.

În acest model, o tabelă este formată din mai multe atribute, iar fiecare linie a tabelei reprezintă un tuplu, adică o instanță concretă a unei entități. Structura relațională se bazează pe conceptele de relație, domeniu, atribut și tuplu. Schema relațională are o importanță majoră, deoarece asigură organizarea clară și eficientă a datelor, menține consistența și integritatea acestora, elimină redundanțele și permite extinderea și administrarea ușoară a sistemului.

2.3. Normalizarea bazei de date:

Încă de la etapa proiectării bazei de date, am avut grijă să respect normele normalizării bazei de date. Am avut grijă ca fiecare câmp, tabel și relație să fie bine structurată. Chiar dacă baza de date pare mică la prima vedere, ascunde după ea o structurare bine definită astfel încât să se poate evita, pe cât de mult posibil, reduntanțele.

PRIMA FORMĂ NORMALĂ:

Prima Formă Normală (1NF) asigură că structura unei tabele dintr-o bază de date este organizată într-un mod care facilitează gestionarea și interogarea acesteia. O relație (tabelă) se consideră a fi în **Prima Formă Normală (1NF)** dacă sunt îndeplinite următoarele condiții:

- Toate atributele (coloanele) conțin doar valori atomice (indivizibile).
- Fiecare coloană conține valori de un singur tip.
- Fiecare înregistrare (rând) este unică, adică poate fi identificată printr-o cheie primară.
- Nu există grupuri care se repetă sau tablouri (liste) în niciun rând.

A DOUA FORMĂ NORMALĂ:

Forma Normală a Doua (2NF) se bazează pe conceptul de dependență funcțională completă. Este o metodă de organizare a unei tabele dintr-o bază de date astfel încât să se reducă redundanța și să se asigure consistența datelor.

Dependența funcțională completă înseamnă că un atribut care nu face parte din cheia primară depinde de întreaga cheie primară, nu doar de o parte a acesteia.

Pentru ca o tabelă să fie în **Forma Normală a Doua (2NF)**, trebuie mai întâi să îndeplinească următoarele condiții:

- Să respecte cerințele **Formei Normale Întâi (1NF)**
- Să elimine **dependențele parțiale**

A TREIA FORMĂ NORMALĂ:

Forma Normală a Treia (3NF) se bazează pe prima (1NF) și a doua (2NF) formă normală. Atingerea acestei forme asigură eliminarea **dependențelor tranzitive**, reducând astfel riscul apariției anomaliilor în date. Deși tabelele aflate în 2NF au mai puțină redundanță decât cele din 1NF, pot exista încă probleme la actualizarea datelor.

O relație se află în **Forma Normală a Treia (3NF)** dacă îndeplinește următoarele condiții:

- Este în **2NF**, adică nu există dependențe parțiale.
- Nu are **dependențe tranzitive** — niciun atribut care nu face parte din cheie nu trebuie să depindă de alt atribut care, la rândul lui, nu face parte din cheie; toate atributele trebuie să depindă direct de cheia primară.

3. Constrângerile de integritate:

Constrângerile SQL sunt utilizate pentru a stabili reguli pentru datele dintr-o tabelă.

Aceste constrângeri limitează tipul de date care pot fi introduse într-o tabelă, asigurând astfel acuratețea și fiabilitatea informațiilor stocate. Dacă există o neconcordanță între o constrângere și acțiunea efectuată asupra datelor, acea acțiune este anulată.

Constrângerile pot fi de **nivel de coloană** sau de **nivel de tabelă.**

3.1. Constrângerea `NOT NULL`:

Asigură că o coloană nu poate avea valoarea **NULL** (adică nu poate rămâne necompletată). De exemplu:

```
CREATE TABLE Tabara_de_zi (
     Ora_de_inceput TIME NOT NULL
);
```

3.2. Constrângerea UNIQUE:

Asigură că toate valorile dintr-o coloană sunt **unice**, fără duplicări. De exemplu:

```
ALTER TABLE Client
ADD CONSTRAINT UQ_Email UNIQUE (Email);
```

3.3. Constrângerea PRIMARY KEY:

Reprezintă o combinație între **NOT NULL** și **UNIQUE**; identifică în mod unic fiecare rând dintr-o tabelă. De exemplu:

```
CREATE TABLE Produs (
     ID_produs INT PRIMARY KEY IDENTITY(1, 1)
);
```

3.4. Constrângerea FOREIGN KEY:

Previne acțiunile care ar distruge legăturile dintre tabele (menține integritatea referențială). De exemplu:

```
ALTER TABLE Magazin
ADD FOREIGN KEY (Servicii) REFERENCES Serviciu(ID_serviciu);
```

3.5. Constrângerea CHECK:

Verifică dacă valorile dintr-o coloană respectă o anumită condiție specificată. De exemplu:

```
CREATE TABLE Client (
    Varsta INT,
    CONSTRAINT chk_varsta CHECK (Varsta >= 18)
);
```

3.6. Constrângerea DEFAULT:

Setează o valoare implicită pentru o coloană atunci când nu este introdusă nicio valoare. De exemplu:

```
CREATE TABLE Client (
    Tara VARCHAR(50) DEFAULT 'Republica Moldova'
);
```

4. Securizarea bazei de date:

Bazele de date au devenit piloni esențiali ai infrastructurii digitale pe care se sprijină multe aplicații moderne. Ele gestionează și asigură integritatea informațiilor critice care susțin atât procesele zilnice ale unei organizații, cât și deciziile sale strategice. Tocmai datorită acestei importanțe, securizarea bazelor de date trebuie tratată ca o responsabilitate majoră. În centrul administrării bazelor de date relaționale se află limbajul SQL, iar o bună înțelegere a principiilor și tehnicilor de securitate SQL este esențială pentru a preveni accesul neautorizat și pentru a reduce riscurile asociate vulnerabilităților de sistem.

4.1. Utilizatori:

În SQL, utilizatorii sunt conturi care pot accesa și opera în baza de date. Fiecărui utilizatori se pot acorda drepturi (citire, modificare, ștergere etc.) pentru a controla accesul la date. Gestionarea utilizatorilor (creare, acordare/retragere privilegii, ștergere) este esențială pentru securitatea bazei de date. De exemplu:

```
CREATE LOGIN alexandrina WITH PASSWORD = 'alexandrina04';

CREATE USER alexandrina_user FOR LOGIN alexandrina;
```

4.2. Rolurile:

În SQL, **rolurile** sunt grupuri de permisiuni care se pot acorda mai ușor utilizatorilor. Ele simplifică gestionarea accesului la date.

4.3. Mascarea informației:

Mascarea datelor în SQL presupune ascunderea parțială a informațiilor sensibile, cum ar fi numerele de card, adresele de e-mail sau parolele, pentru a proteja confidențialitatea. De exemplu, un număr de card poate fi afișat ca `**** **** **** 1234`. Astfel, utilizatorii fără permisiuni speciale pot vedea doar o parte din date, fără a avea acces la valorile complete.

4.4. Vederile:

O **vedere** în SQL este o tabelă virtuală bazată pe rezultatul unei interogări salvate, care permite utilizatorilor să simplifice interogările complexe și să sporească securitatea prin restricționarea accesului la datele din tabelele de bază. View-urile **nu stochează date** propriu-zise, ele generează datele în mod dinamic de fiecare dată când sunt accesate. De exemplu:

```
CREATE VIEW AnimalulDeCompanie AS
```

```
    SELECT c.Nume + c.Prenume AS Clientul, a.Nume AS
AnimalulDeCompanie
    FROM Client c
    INNER JOIN Animal a ON a.Stapan = C.ID_client;


SELECT * FROM AnimalulDeCompanie;
```

4.5. Indecși:

Indecșii în SQL sunt structuri speciale de date care îmbunătățesc viteza operațiunilor de căutare, permițând bazei de date să găsească rapid rândurile necesare fără a parcurge întreaga tabelă. Ele pot fi create pe una sau mai multe coloane și ajută la creșterea performanței interogărilor, însă un număr prea mare de indexuri poate încetini operațiunile de modificare a datelor, cum ar fi **INSERT**, **UPDATE** sau **DELETE**. De exemplu:

```
CREATE INDEX idx_nume_animal
ON Animal(Nume);

SELECT Nume FROM Animal WHERE Nume LIKE 'B%';
```

4.6. Criptarea informației:

Criptarea datelor în SQL este procesul de transformare a informațiilor sensibile dintr-o bază de date într-un format inaccesibil, folosind algoritmi criptografici. Doar utilizatorii autorizați cu cheia corespunzătoare pot accesa datele originale. Aceasta protejează datele atât în repaus, cât și în tranzit, făcându-le inutilizabile în caz de acces neautorizat.

Criptarea datelor poate fi realizată în majoritatea dialectelor SQL moderne, dar modul de implementare diferă între ele:

- **SQL Server** – suportă **Transparent Data Encryption (TDE)**, **Always Encrypted** și funcții de criptare precum `ENCRYPTBYKEY`, `ENCRYPTBYPASSPHRASE`.

- **MySQL / MariaDB** – suportă criptarea prin funcții precum `AES_ENCRYPT()` și `AES_DECRYPT()`.
- **PostgreSQL** – poate folosi extensii precum `pgcrypto`.
- **Oracle** – oferă **Transparent Data Encryption (TDE)** și funcții de criptare cum ar fi `DBMS_CRYPTO`.
- **SQLite** – suportă criptare prin extensii externe, de exemplu `SQLCipher`.

4.7. Schemele:

O **schemă a bazei de date** definește **structura** și **organizarea** datelor într-o bază de date. Aceasta arată cum sunt stocate datele în mod logic, inclusiv relațiile dintre diferite tabele și alte obiecte ale bazei de date. Schema funcționează ca un plan pentru modul în care datele sunt **stocate**, **accesate** și **manipulate**, asigurând **consistența și integritatea întregului sistem**. De exemplu:

```
CREATE SCHEMA Creare

CREATE VIEW Creare.PC AS
SELECT Cod, Model, Pret FROM pc_uri

SELECT * FROM Creare.PC
```

4.8. Sinonimele:

Un **sinonim** oferă un alt nume pentru un obiect din baza de date, numit obiect original, care poate exista fie pe serverul local, fie pe un alt server. Un sinonim aparține unei scheme, iar numele sinonimului trebuie să fie unic. Un sinonim **nu poate fi obiect original pentru un alt sinonim** și nu poate face referire la o funcție definită de utilizator. De exemplu:

```
CREATE SYNONYM informatii_produse
FOR dbo.produse
```

5. Concluzie:

În cadrul acestui studiu individual am realizat nu doar baza de date a unei mici companii, dar, totodată, m-am asigurat că baza de date este securizată la un nivel de bază. Mi-am antrenat abilitățile de a crea o bază de date, de la structură, la diagramă, la cod. Pe lângă aceasta, am învățat lucruri noi, ca de exemplu lucrul cu indecșii, sinonimele și userii într-un mod mai aprofundat. Acest studiu individual a reușit să-mi dezvolte gândirea logică în ceea ce ține bazele de date, structurarea acesteia și lucrul cu datele, până și chiar optimizarea interogărilor. Pe parcursul lucrării, cel mai greu mi-a fost să lucrez cu criptarea datelor, din cauza codului și logicii complexe din spatele acesteea, dar și cu schemele, însă cu ajutorul documentației am reușit să realizez cu succes sarcina. Cea mai bună parte a acestui studiu a fost faptul că am lucrat cu baza de date unei companii plăcute, procesul de combinare a tehnologiei cu viața de zi cu zi devenind nu doar mai interesant, dar și totodată arătând adevărata importanță a bazelor de date și cum ne ajută ele, chiar și în cele mai banale locuri.

6. Webografie

1. GeeksforGeeks - informații despre sinonime
2. GeeksforGeeks - informații despre scheme
3. GeeksforGeeks - informații despre criptarea datelor
4. W3Schools - tutorial SQL
5. GeeksforGeeks - normalizarea bazei de date
6. Cursul “Implementarea Limbajului SQL” - Moodle
7. Cursul “Sistemele de Gestiune a Bazelor de Date” - Moodle
8. Draw.io - crearea disgramelor

7. Anexe:

1. GitHub Repository cu codul
2. Link către Drive