Proiect Baze de Date

Baza de date a unui magazin alimentar

- Se doreşte informatizarea activităţii unui magazin alimentar.

Se consideră următoarea colecţie a datelor:

Cod medicament, Denumire medicament, Pret, Nr. bon casă, Dată bon casă, Cantitate vândută, Cod firma, Nume firma, Cod categorie, Denumire categorie, Cod client, Nume client, Telefon,

Reguli de gestiune:

-un client este o persoană care are cel puțin o achiziție;
-pentru fiecare client se reține datele sale personale(nume și numărul de telefon);
-pentru fiecare comandă se reține una sau mai multe vanzări și clientul;
-pentru fiecare produs se reține denumirea, categoria și producătorul;

1.Modelul Conceptual:

Entitati:

1.Produs

Atribute: id, produs, pret

Identificator: id

2.Bon

Atribute: bon, data

Identificator: bon

3.Client

Atribute: id, nume, telefon

Identificator: id

4.Firma:

Atribute: id, firma

Identificator: firma

5.Categorie:

Atribute: id, categorie

Identificator: categorie

Asocieri intre entitati:

1.Bon – Produs:

Cardinalitate: n:n

Un produs se poate afla pe mai multe bonuri si mai multe bonuri pot avea mai multe produse.

2.Bon – Client:

Cardinalitate: n:1

Un client poate avea mai multe bonuri.

3.Produs – Categorie:

Cardinalitate: n:1

O categorie poate include mai multe produse.

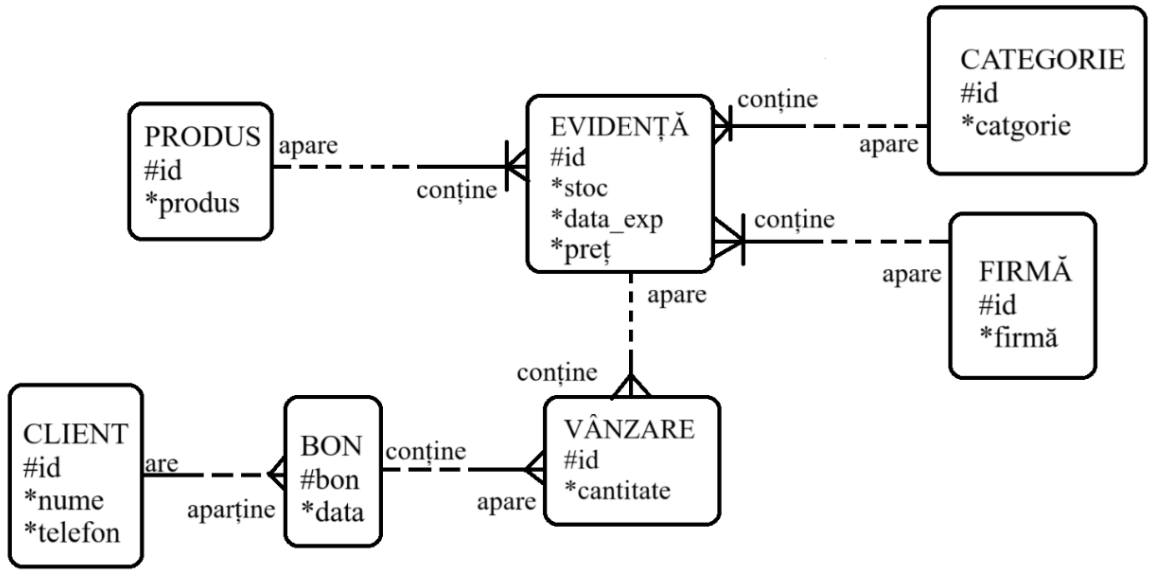
4.Produs – Firma:

Cardinalitate: n:n

O firma poate produce mai multe produse, iar acelasi produs poate fi fabricat de mai multe firme

Rezolvarea relatiilor de tip n:n :

- Intre Produs si Bon apare tabela Vanzare cu id si cantitate.
- Pentru a rezolva relatia n:n intre Produs si Firma apare tabela Evident ce va prelua pretul din tabela produs si vor aparea id, stoc si data_exp.



Colul SQL pentru:

1) CREAREA TABELELOR:

- Tabela Clientilor

```
CREATE TABLE Clienti (
  – clientID CHAR(10) CONSTRAINT pk_Clienti PRIMARY KEY,
  – nume VARCHAR(50),
  – telefon VARCHAR(12),
);
```

- Tabela Categoriilor

```
CREATE TABLE Categorii (
  – categorieID CHAR(10) CONSTRAINT pk_Categorii PRIMARY KEY,
  – categorie VARCHAR(50)
);
```

- Tabela Bonurilor

```
CREATE TABLE Bonuri  (
  – bonID CHAR(10) CONSTRAINT pk_Bonuri PRIMARY KEY,
  – data DATE,
  – clientID CHAR(10) CONSTRAINT fk_Clienti FOREIGN KEY
    REFERENCES Clienti(clientID)
  );
```

- Tabela Produselor

```
CREATE TABLE Produse (
  – produsID CHAR(10) CONSTRAINT pk_Produse PRIMARY KEY,
  – produs VARCHAR(50) );
```

- Tabela Firmelor

CREATE TABLE Firme (

- firmaID CHAR(10) CONSTRAINT pk_Firme PRIMARY KEY,
- firma VARCHAR(50)

);

- Tabela Vanzarilor:

CREATE TABLE Vanzari (

- vanzariID CHAR(10) CONSTRAINT pk_Vanzari PRIMARY KEY,
- bonID CHAR(10) CONSTRAINT fk_Bonuri FOREIGN KEY REFERENCES Bonuri(bonID),
- produsID CHAR(10) CONSTRAINT fk_Produse FOREIGN KEY REFERENCES Produse(produsID),
- cantitate INT

);

- Tabela Evidentelor:

CREATE TABLE Evidente (

- evidentaID CHAR(10) CONSTRAINT pk_Evidente PRIMARY KEY,
- categorieID CHAR(10) CONSTRAINT fk_Categorii FOREIGN KEY REFERENCES Categorii (categorieID),
- firmaID CHAR(10) CONSTRAINT fk_Firme FOREIGN KEY REFERENCES Firme (firmaID),
- produsID CHAR(10) CONSTRAINT fk_Produse FOREIGN KEY REFERENCES Produse(produsID),
- stoc INT,
- pret INT,
- dataExp DATE,

);

2) MODIFICAREA STRUCTURII TABELELOR:

```
ALTER TABLE Clienti
ADD email VARCHAR(100);
```

3) MANIPULAREA DATELOR (INSERT, UPDATE, DELETE):

```
INSERT INTO Clienti (clientID, nume, telefon, email)
VALUES ('C001', 'Popescu Ion', '0712345678', 'ion.popescu@email.com');

UPDATE Clienti
SET telefon = '0722334455'
WHERE clientID = 'C001';

DELETE FROM Clienti
WHERE clientID = 'C001';

INSERT INTO Clienti (clientID, nume, telefon, email) VALUES
('C001', 'Ion Popescu', '0711223344', 'ion.popescu@gmail.com'),
('C002', 'Maria Ionescu', '0722334455', 'maria.ionescu@yahoo.com'),
('C003', 'George Vasilescu', '0733445566', 'george.vasilescu@outlook.com');

INSERT INTO Categorii (categorieID, categorie) VALUES
('CAT01', 'Electronice'),
```

('CAT02', 'Electrocasnice'),

('CAT03', 'IT&C');

INSERT INTO Firme (firmaID, firma) VALUES

('F001', 'Samsung'),

('F002', 'Philips'),

('F003', 'Dell');

INSERT INTO Produse (produsID, produs) VALUES

('P001', 'Televizor LED 4K'),

('P002', 'Frigider No Frost'),

('P003', 'Laptop Inspiron');

INSERT INTO Evidente (evidentaID, categorieID, firmaID, produsID, stoc, pret, dataExp) VALUES

('E001', 'CAT01', 'F001', 'P001', 20, 2000, '2026-12-31'),

('E002', 'CAT02', 'F002', 'P002', 15, 1500, '2027-01-15'),

('E003', 'CAT03', 'F003', 'P003', 10, 3500, '2025-11-30');

INSERT INTO Bonuri (bonID, data, clientID) VALUES

('B001', '2025-05-01', 'C001'),

('B002', '2025-05-02', 'C002'),

('B003', '2025-05-03', 'C003');

INSERT INTO Vanzari (vanzariID, bonID, produsID, cantitate) VALUES

('V001', 'B001', 'P001', 1),

('V002', 'B001', 'P003', 1),

('V003', 'B002', 'P002', 2),

('V004', 'B003', 'P001', 1);

4) SELECT

```
SELECT * FROM Produse;

SELECT data FROM Bonuri;
```

5) SELECTAREA DATELOR DIN MAI MULTE TABELE (JOIN):

```
SELECT c.nume, b.data, v.cantitate

FROM Vanzari v

JOIN Bonuri b ON v.bonID = b.bonID

JOIN Clienti c ON b.clientID = c.clientID;
```

6) SUBINTEROGARI (necorelate si corelate):

```
SELECT produsID

FROM Evidente

WHERE pret > (SELECT AVG(pret) FROM Evidente);

SELECT DISTINCT c.nume
```

```
FROM Clienti c
WHERE EXISTS (
   SELECT 1
   FROM Bonuri b
   WHERE b.clientID = c.clientID
   AND (SELECT COUNT(*) FROM Vanzari v WHERE v.bonID = b.bonID) > 1
);
```

7) SINTEZA DATELOR (agreagare)

```
SELECT produsID, SUM(cantitate) AS total_vandut
FROM Vanzari
GROUP BY produsID;

SELECT clientID, COUNT(bonID) AS numar_bonuri
FROM Bonuri
GROUP BY clientID;
SELECT c.categorie, AVG(e.pret) AS pret_mediu
FROM Evidente e
JOIN Categorii c ON e.categorieID = c.categorieID
GROUP BY c.categorie
HAVING AVG(e.pret) > 2000;

SELECT c.categorie, MAX(e.pret) AS pret_maxim
```

```
FROM Evidente e
JOIN Categorii c ON e.categorieID = c.categorieID
GROUP BY c.categorie;
```

8) UTILIZAREA VEDERILOR

```
CREATE VIEW V_VanzariDetaliate AS
SELECT
    v.vanzariID,
    c.nume AS nume_client,
    p.produs,
    v.cantitate,
    b.data AS data_vanzare
FROM Vanzari v
JOIN Bonuri b ON v.bonID = b.bonID
JOIN Clienti c ON b.clientID = c.clientID
JOIN Produse p ON v.produsID = p.produsID;
```

Vanzarile din ultimele 7 zile:

```
SELECT *
FROM V_VanzariDetaliate
WHERE data_vanzare >= DATEADD(DAY, -7, GETDATE());
```

9) INDEXAREA TABELELOR

```
CREATE NONCLUSTERED INDEX idx_Clienti_nume
ON Clienti(nume);

SELECT * FROM Clienti WHERE nume LIKE 'Ion%';
```

10) CREAREA FUNCTIILOR SI PROCEDURILOR STOCATE

```
CREATE FUNCTION dbo.GetPretProdus(@produsID CHAR(10))
RETURNS INT
AS
BEGIN
    DECLARE @pret INT;
    SELECT @pret = pret FROM Evidente WHERE produsID = @produsID;
    RETURN @pret;
END;

CREATE PROCEDURE dp_AdaugaBon
    @bonID CHAR(10),
    @data DATE,
    @clientID CHAR(10)
AS
BEGIN
```

```
    INSERT INTO Bonuri (bonID, data, clientID)
    VALUES (@bonID, @data, @clientID);
END;
```

11) TRANZACTII SQL

```
BEGIN TRANSACTION;

UPDATE Evidente
SET stoc = stoc - 2
WHERE produsID = 'P001';

INSERT INTO Vanzari (vanzariID, bonID, produsID, cantitate)
VALUES ('V001', 'B001', 'P001', 2);

COMMIT;
```

12) CURSOARE

```
DECLARE @produsID CHAR(10), @stoc INT;
DECLARE produs_cursor CURSOR FOR
SELECT produsID, stoc FROM Evidente;

OPEN produs_cursor;
```

```
FETCH NEXT FROM produs_cursor INTO @produsID, @stoc;
WHILE @@FETCH_STATUS = 0
BEGIN
    PRINT 'ProdusID: ' + @produsID + ' - Stoc: ' + CAST(@stoc AS VARCHAR);
    FETCH NEXT FROM produs_cursor INTO @produsID, @stoc;
END;
CLOSE produs_cursor;
DEALLOCATE produs_cursor;
```

13) TRIGGERE

```
CREATE TRIGGER trg_VerificaStoc
ON Evidente
AFTER UPDATE
AS
BEGIN
    IF EXISTS (
        SELECT * FROM inserted WHERE stoc < 0
    )
    BEGIN
        RAISERROR ('Stocul nu poate fi negativ!', 16, 1);
        ROLLBACK TRANSACTION;
    END
END;
```