Aplicatie APEX:

<https://apex.oracle.com/pls/apex/r/raton/sgbd-2226739/home?session=1393499055946>

1. Descrierea bazei de date

Proiectul are ca scop gestiunea comenzilor online. Acest proiect este realizat in apex oracle. Proiectul contine notiuni fundamentale de baze de date, descrierea lucrarii, tabele, atribute, relatii, restrictii, schema tabelelor, codul SQL pentru tabele si comenzi sql.

Acesta baza de date contine informatii despre clienti (id, nume, prenume, telefon, adresa), comenzi (id, data, modalitate, id-ul clientului), produse (id, denumire, descriere, pret) si stocul lor. Tabela produse\_comenzi face legatura dintre Produse si Comenzi.

2. Schema bazei de date

A diagram of a product

Description automatically generated

set serveroutput on;

--1.Afisam produsele cu pretul unitar mai mic sau egal cu 100 si cantitatea disponibila din stoc

DECLARE

CURSOR c1 IS

SELECT p.id\_produs, p.denumire\_produs, p.pret\_unitar, s.cantitate\_stoc

FROM ProduseNet p

INNER JOIN StocNet s ON p.id\_produs = s.id\_produs;

v\_id\_produs ProduseNet.id\_produs%TYPE;

v\_denumire\_produs ProduseNet.denumire\_produs%TYPE;

v\_pret\_unitar ProduseNet.pret\_unitar%TYPE;

v\_cantitate\_stoc StocNet.cantitate\_stoc%TYPE;

BEGIN

OPEN c1;

LOOP

FETCH c1 INTO v\_id\_produs, v\_denumire\_produs, v\_pret\_unitar, v\_cantitate\_stoc;

EXIT WHEN c1%NOTFOUND;

IF v\_pret\_unitar <= 100 THEN

DBMS\_OUTPUT.PUT\_LINE('Produsul ' || v\_denumire\_produs || ' are un pret de ' || v\_pret\_unitar || ' si o cantitate disponibila în stoc de ' || v\_cantitate\_stoc);

END IF;

END LOOP;

CLOSE c1;

END;

/

A screenshot of a computer program

Description automatically generated

--2. Acesta este blocul PL/SQL actualizat pentru afișarea primelor 4 comenzi cu cele mai multe produse

DECLARE

CURSOR c2 IS

SELECT c.id\_comanda, c.data\_comanda, COUNT(pc.id\_produs) AS nr\_produse

FROM ComenziNet c

INNER JOIN Produse\_comenzi pc ON c.id\_comanda = pc.id\_comanda

GROUP BY c.id\_comanda, c.data\_comanda

ORDER BY nr\_produse DESC;

v\_rec c2%ROWTYPE;

BEGIN

IF NOT c2%ISOPEN THEN

OPEN c2;

END IF;

FOR i IN 1..4 LOOP

FETCH c2 INTO v\_rec;

EXIT WHEN c2%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('Comanda ' || v\_rec.id\_comanda || ' a fost plasata la data de ' || v\_rec.data\_comanda || ' si contine ' || v\_rec.nr\_produse || ' produse');

END LOOP;

CLOSE c2;

END;

/

A screenshot of a computer program

Description automatically generated

--3.Sa se stearga din tabela comenzi toate comenzile plasate printr-o modalitate introdusa prin de la tastatura. Afisati cate comenzi au fost sterse.

ACCEPT g\_modalitate PROMPT 'Introduceți modalitatea de plasare a comenzii'

VARIABLE g\_nrcomenzi NUMBER;

BEGIN

DELETE FROM Produse\_comenzi pc

WHERE pc.id\_comanda IN (SELECT cn.id\_comanda FROM ComenziNet cn WHERE cn.modalitate = '&g\_modalitate');

DELETE FROM ComenziNet WHERE modalitate = '&g\_modalitate';

:g\_nrcomenzi := SQL%ROWCOUNT;

DBMS\_OUTPUT.PUT\_LINE('Numărul de comenzi eliminate: ' || NVL(TO\_CHAR(:g\_nrcomenzi), '0'));

END;

/

SET SERVEROUTPUT ON

PRINT g\_nrcomenzi;

ROLLBACK;

A screenshot of a computer

Description automatically generated

--4. Afisare produs comandat, data comenzii, metoda de plata

DECLARE

v\_nume\_produs ProduseNet.denumire\_produs%TYPE;

v\_data\_comanda ComenziNet.data\_comanda%TYPE;

v\_modalitate\_comanda ComenziNet.modalitate%TYPE;

BEGIN

FOR comanda IN (SELECT p.denumire\_produs, c.data\_comanda, c.modalitate

FROM ComenziNet c

INNER JOIN Produse\_comenzi pc ON c.id\_comanda = pc.id\_comanda

INNER JOIN ProduseNet p ON pc.id\_produs = p.id\_produs) LOOP

v\_nume\_produs := comanda.denumire\_produs;

v\_data\_comanda := comanda.data\_comanda;

v\_modalitate\_comanda := comanda.modalitate;

DBMS\_OUTPUT.PUT\_LINE('Produsul ' || v\_nume\_produs || ' a fost comandat în data de ' || TO\_CHAR(v\_data\_comanda, 'DD-MM-YYYY') || ' cu metoda de plata ' || v\_modalitate\_comanda);

END LOOP;

END;

/

A screenshot of a computer

Description automatically generated

--5 Detalii despre comenzile plasate (data, client, cantitate, valoare totala comanda). În plus, comenzile să fie ordonate descrescător în funcție de valoarea totală a comenzii.

DECLARE

CURSOR c5 IS

SELECT c.id\_comanda, c.data\_comanda, cl.nume || ' ' || cl.prenume AS nume\_client,

SUM(pc.cantitate) AS cantitate, SUM(pc.cantitate \* p.pret\_unitar) AS total\_valoare

FROM ComenziNet c

INNER JOIN Produse\_comenzi pc ON c.id\_comanda = pc.id\_comanda

INNER JOIN ProduseNet p ON pc.id\_produs = p.id\_produs

INNER JOIN ClientiNet cl ON c.id\_client = cl.id\_client

GROUP BY c.id\_comanda, c.data\_comanda, cl.nume, cl.prenume

ORDER BY total\_valoare DESC;

v\_id\_comanda ComenziNet.id\_comanda%TYPE;

v\_data\_comanda ComenziNet.data\_comanda%TYPE;

v\_nume\_client VARCHAR2(100);

v\_cantitate NUMBER;

v\_total\_valoare NUMBER;

BEGIN

OPEN c5;

LOOP

FETCH c5 INTO v\_id\_comanda, v\_data\_comanda, v\_nume\_client, v\_cantitate, v\_total\_valoare;

EXIT WHEN c5%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('Comanda cu ID ' || v\_id\_comanda || ' a fost plasata la data de ' || TO\_CHAR(v\_data\_comanda, 'DD-MM-YYYY') ||

' de catre ' || v\_nume\_client || ' care contine ' || v\_cantitate || ' produse in valoare de ' || v\_total\_valoare || ' lei.');

END LOOP;

CLOSE c5;

END;

/

A screenshot of a computer

Description automatically generated

/\*6. Actualizați prețul de listă al produselor în funcție de valoarea minimă a prețului unitar, astfel

- produsele cu prețul unitar între 1 și 100 să primească o majorare de 20%,

- produsele cu prețul unitar între 101 și 1000 să primească o majorare de 10%.

--Afișați numărul total de modificări efectuate.\*/

DECLARE

v\_nrmodificari NUMBER := 0;

CURSOR c6 IS SELECT id\_produs, pret\_unitar FROM ProduseNet;

BEGIN

FOR produs IN c6 LOOP

IF produs.pret\_unitar >= 1 AND produs.pret\_unitar <= 100 THEN

UPDATE ProduseNet

SET pret\_unitar = ROUND(produs.pret\_unitar \* 1.2, 2) -- marire cu 20%

WHERE id\_produs = produs.id\_produs;

v\_nrmodificari := v\_nrmodificari + 1;

ELSIF produs.pret\_unitar >= 101 AND produs.pret\_unitar <= 1000 THEN

UPDATE ProduseNet

SET pret\_unitar = ROUND(produs.pret\_unitar \* 1.1, 2) -- marire cu 10%

WHERE id\_produs = produs.id\_produs;

v\_nrmodificari := v\_nrmodificari + 1;

END IF;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Numarul total de modificari efectuate: ' || v\_nrmodificari);

END;

/

rollback;

A screenshot of a computer

Description automatically generated

--7. Bloc PL/SQL pentru a insera o comandă noua

BEGIN

INSERT INTO ComenziNet (id\_comanda, data\_comanda, modalitate, id\_client)

VALUES (9, SYSDATE, 'card', 1);

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX THEN

DBMS\_OUTPUT.PUT\_LINE('Comanda cu acest ID exista deja in baza de date.');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('A aparut o alta eroare');

END;

/

A screenshot of a computer

Description automatically generated

--8. Bloc PL/SQL pentru a mari pretul unui produs cu 10%

DECLARE

produs\_not\_found EXCEPTION;

BEGIN

UPDATE ProduseNet

SET pret\_unitar = pret\_unitar \* 1.1

WHERE id\_produs = &p\_id;

IF SQL%NOTFOUND THEN

RAISE produs\_not\_found;

ELSE

DBMS\_OUTPUT.PUT\_LINE('Pretul produsului a fost marit');

END IF;

EXCEPTION

WHEN produs\_not\_found THEN

DBMS\_OUTPUT.PUT\_LINE('Produsul cu ID-ul specificat nu exista in baza de date.');

END;

/

ROLLBACK;

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

/\*9. Bloc PL/SQL care să afișeze detaliile comenzilor înregistrate dintr-un an citit de la tastatură din tabela ComenziNet.

În cazul în care sunt găsite mai multe comenzi, nu sunt gasite deloc se va afisa un mesaj\*/

DECLARE

v\_id\_comanda ComenziNet.id\_comanda%TYPE;

v\_data\_comanda ComenziNet.data\_comanda%TYPE;

v\_modalitate ComenziNet.modalitate%TYPE;

v\_cantitate\_produse ComenziNet.cantitate\_produse%TYPE;

v\_an NUMBER := &an;

BEGIN

SELECT id\_comanda, data\_comanda, modalitate, cantitate\_produse

INTO v\_id\_comanda, v\_data\_comanda, v\_modalitate, v\_cantitate\_produse

FROM ComenziNet

WHERE EXTRACT(YEAR FROM data\_comanda) = v\_an;

DBMS\_OUTPUT.PUT\_LINE('Comenzi din anul ' || v\_an || ':');

DBMS\_OUTPUT.PUT\_LINE('ID Comanda: ' || v\_id\_comanda || ', Data Comanda: ' || v\_data\_comanda || ', Modalitate: ' || v\_modalitate || ', Cantitate Produse: ' || v\_cantitate\_produse);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('In anul ' || v\_an || ' nu a fost inregistrata nicio comanda.');

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('In anul ' || v\_an || ' au fost inregistrate mai multe comenzi.');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('A aparut o alta problema: ' || SQLERRM);

END;

/

A computer screen shot of a program

Description automatically generated

/\*10.Verifică dacă există produse cu prețul mai mare de 20 în stoc. Dacă da, afișează numărul de produse care îndeplinesc această condiție,

altfel afișează un mesaj care indică că nu există produse cu prețul mai mare de 20 în stoc.\*/

DECLARE

v\_count NUMBER;

v\_exception EXCEPTION;

BEGIN

SELECT COUNT(\*)

INTO v\_count

FROM StocNet s

INNER JOIN ProduseNet p ON s.id\_produs = p.id\_produs

WHERE p.pret\_unitar > 20;

IF v\_count > 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Exista ' || v\_count || ' produse cu pretul mai mare de 20 in stoc.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Nu exista produse cu pretul mai mare de 20 in stoc.');

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista produse in stoc.');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('A aparut o eroare: ' || SQLERRM);

END;

/

A screenshot of a computer

Description automatically generated

/\*11. Pachet cu: adaugare produs nou in tabela produsenet, modifica categoria produsului, afisare categorie si descriere completa a produsului\*/

CREATE OR REPLACE PACKAGE Pachet\_Produse AS

PROCEDURE Adauga\_Produs(

id\_produs IN NUMBER,

denumire\_produs IN VARCHAR2,

descriere IN VARCHAR2,

pret\_unitar IN NUMBER,

categorie IN VARCHAR2

);

PROCEDURE Modifica\_Categorie\_Produs(

id\_produs IN NUMBER,

noua\_categorie IN VARCHAR2

);

FUNCTION Categorie\_Produs(

id\_produs IN NUMBER

) RETURN VARCHAR2;

FUNCTION Descriere\_Completa\_Produs(

id\_produs IN NUMBER

) RETURN VARCHAR2;

END Pachet\_Produse;

/

CREATE OR REPLACE PACKAGE BODY Pachet\_Produse AS

PROCEDURE Adauga\_Produs(

id\_produs IN NUMBER,

denumire\_produs IN VARCHAR2,

descriere IN VARCHAR2,

pret\_unitar IN NUMBER,

categorie IN VARCHAR2

) IS

produs\_existent EXCEPTION;

PRAGMA EXCEPTION\_INIT(produs\_existent, -00001);

BEGIN

INSERT INTO ProduseNet (id\_produs, denumire\_produs, descriere, pret\_unitar, categorie)

VALUES (id\_produs, denumire\_produs, descriere, pret\_unitar, categorie);

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Inregistrare adaugata cu succes in tabela ProduseNet.');

EXCEPTION

WHEN produs\_existent THEN

DBMS\_OUTPUT.PUT\_LINE('Exista deja un produs cu id-ul specificat in tabela ProduseNet.');

END Adauga\_Produs;

PROCEDURE Modifica\_Categorie\_Produs(

id\_produs IN NUMBER,

noua\_categorie IN VARCHAR2

) IS

produs\_negasit EXCEPTION;

BEGIN

UPDATE ProduseNet SET categorie = noua\_categorie WHERE id\_produs = id\_produs;

IF SQL%ROWCOUNT = 0 THEN

RAISE produs\_negasit;

ELSE

DBMS\_OUTPUT.PUT\_LINE('Categoria produsului a fost modificată cu succes.');

END IF;

EXCEPTION

WHEN produs\_negasit THEN

DBMS\_OUTPUT.PUT\_LINE('Produsul cu id-ul specificat nu a fost găsit în tabela ProduseNet.');

END Modifica\_Categorie\_Produs;

FUNCTION Categorie\_Produs(

id\_produs IN NUMBER

) RETURN VARCHAR2 IS

categoria\_produs VARCHAR2(100);

BEGIN

SELECT categorie INTO categoria\_produs FROM ProduseNet WHERE id\_produs = id\_produs;

RETURN categoria\_produs;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN 'Produs inexistent';

WHEN TOO\_MANY\_ROWS THEN

RETURN 'Mai multe categorii găsite';

END Categorie\_Produs;

FUNCTION Descriere\_Completa\_Produs(

id\_produs IN NUMBER

) RETURN VARCHAR2 IS

descriere VARCHAR2(1000);

BEGIN

SELECT descriere INTO descriere FROM ProduseNet WHERE id\_produs = id\_produs;

RETURN descriere;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN 'Produs inexistent';

WHEN TOO\_MANY\_ROWS THEN

RETURN 'Mai multe descrieri găsite';

END Descriere\_Completa\_Produs;

END Pachet\_Produse;

/

--apel Adauga\_Produs

BEGIN

Pachet\_Produse.Adauga\_Produs(11, 'Scaun', 'Scaun ergonimic', 300, 'Mobilier');

END;

/

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

--apel Modifica\_Categorie\_Produs

BEGIN

Pachet\_Produse.Modifica\_Categorie\_Produs(11, 'mobila');

END;

/

A screenshot of a computer

Description automatically generated

--apel Categorie\_Produs

DECLARE

categoria\_produs VARCHAR2(20);

BEGIN

categoria\_produs := Pachet\_Produse.Categorie\_Produs(11);

DBMS\_OUTPUT.PUT\_LINE('Categoria produsului este: ' || categoria\_produs);

END;

/

A screenshot of a computer

Description automatically generated

--apel Descriere\_Completa\_Produs

DECLARE

descriere\_completa VARCHAR2(30);

BEGIN

descriere\_completa := Pachet\_Produse.Descriere\_Completa\_Produs(11);

DBMS\_OUTPUT.PUT\_LINE('Descrierea completa a produsului este: ' || descriere\_completa);

END;

/

A screenshot of a computer screen

Description automatically generated

/\*12. Un pachet in care schimbam modalitatea comenzii din cash in ramburs\*/

CREATE OR REPLACE PACKAGE Pachet\_ModificareModalitate AS

PROCEDURE Modifica\_Modalitate\_Comanda(D

id\_comanda IN NUMBER,

noua\_modalitate IN VARCHAR2

);

END Pachet\_ModificareModalitate;

/

CREATE OR REPLACE PACKAGE BODY Pachet\_ModificareModalitate AS

PROCEDURE Modifica\_Modalitate\_Comanda(

id\_comanda IN NUMBER,

noua\_modalitate IN VARCHAR2

) IS

modalitate\_invalida EXCEPTION;

BEGIN

IF noua\_modalitate = 'Cash' THEN

RAISE modalitate\_invalida;

END IF;

UPDATE ComenziNet SET modalitate = noua\_modalitate WHERE id\_comanda = id\_comanda;

DBMS\_OUTPUT.PUT\_LINE('Modalitatea comenzii a fost schimbata cu succes in: ' || noua\_modalitate);

EXCEPTION

WHEN modalitate\_invalida THEN

DBMS\_OUTPUT.PUT\_LINE('Nu se poate schimba modalitatea în "Cash".');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('A intervenit o eroare. Modalitatea comenzii nu a fost modificcata.');

END Modifica\_Modalitate\_Comanda;

END Pachet\_ModificareModalitate;

/

--apel ModificareModalitate

BEGIN

Pachet\_ModificareModalitate.Modifica\_Modalitate\_Comanda(1, 'Ramburs');

END;

/

A screenshot of a computer error

Description automatically generated

/\*13. Pachet cu înregistrarea unei noi comenzi și actualizarea cantității stocului unui produs.

calcularea totalului unei comenzi și verificarea disponibilității în stoc a unui produs.\*/

CREATE OR REPLACE PACKAGE Pachet\_Utilitare IS

PROCEDURE InregistreazaComanda(

id\_comanda IN NUMBER,

data\_comanda IN DATE,

modalitate IN VARCHAR2,

id\_client IN NUMBER

);

FUNCTION CalculTotalComanda(id\_comanda IN NUMBER) RETURN NUMBER;

PROCEDURE ActualizeazaStoc(

id\_produs IN NUMBER,

noua\_cantitate IN NUMBER

);

FUNCTION VerificaDisponibilitateStoc(id\_produs IN NUMBER) RETURN NUMBER;

END Pachet\_Utilitare;

/

CREATE OR REPLACE PACKAGE BODY Pachet\_Utilitare IS

PROCEDURE InregistreazaComanda(

id\_comanda IN NUMBER,

data\_comanda IN DATE,

modalitate IN VARCHAR2,

id\_client IN NUMBER

) IS

BEGIN

INSERT INTO ComenziNet (id\_comanda, data\_comanda, modalitate, id\_client)

VALUES (id\_comanda, data\_comanda, modalitate, id\_client);

COMMIT;

END InregistreazaComanda;

FUNCTION CalculTotalComanda(id\_comanda IN NUMBER) RETURN NUMBER IS

total NUMBER := 0;

BEGIN

SELECT SUM(cantitate \* pret) INTO total FROM Produse\_comenzi WHERE id\_comanda = id\_comanda;

RETURN total;

END CalculTotalComanda;

PROCEDURE ActualizeazaStoc(

id\_produs IN NUMBER,

noua\_cantitate IN NUMBER

) IS

BEGIN

UPDATE StocNet SET cantitate\_stoc = noua\_cantitate WHERE id\_produs = id\_produs;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Stocul pentru produsul cu ID-ul ' || id\_produs || ' a fost actualizat la ' || noua\_cantitate);

END ActualizeazaStoc;

FUNCTION VerificaDisponibilitateStoc(id\_produs IN NUMBER) RETURN NUMBER IS

disponibil NUMBER := 0;

BEGIN

FOR rec IN (SELECT cantitate\_stoc FROM StocNet WHERE id\_produs = id\_produs) LOOP

disponibil := rec.cantitate\_stoc;

EXIT;

END LOOP;

RETURN disponibil;

END VerificaDisponibilitateStoc;

END Pachet\_Utilitare;

/

--apel InregistreazaComanda

BEGIN

Pachet\_Utilitare.InregistreazaComanda(11, SYSDATE, 'card', 1);

END;

/

A computer screen with white text

Description automatically generated

--apel CalculTotalComanda

DECLARE

total\_comanda NUMBER;

BEGIN

total\_comanda := Pachet\_Utilitare.CalculTotalComanda(101);

DBMS\_OUTPUT.PUT\_LINE('Totalul comenzii este: ' || total\_comanda);

END;

/

A screen shot of a computer

Description automatically generated

--apel ActualizeazaStoc

BEGIN

Pachet\_Utilitare.ActualizeazaStoc(1, 50);

END;

/

A screenshot of a computer

Description automatically generated

--apel VerificaDisponibilitateStoc

DECLARE

disponibilitate NUMBER;

BEGIN

disponibilitate := Pachet\_Utilitare.VerificaDisponibilitateStoc(1);

IF disponibilitate > 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Produsul este disponibil în stoc.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Produsul nu este disponibil în stoc.');

END IF;

END;

/

A screenshot of a computer error

Description automatically generated