De Gherasim Rareș
Facultatea de Matematică și Informatică, Specializarea Informatică, Anul II, Grupa 243

Proiect Sisteme de Gestiune ale Bazelor de Date

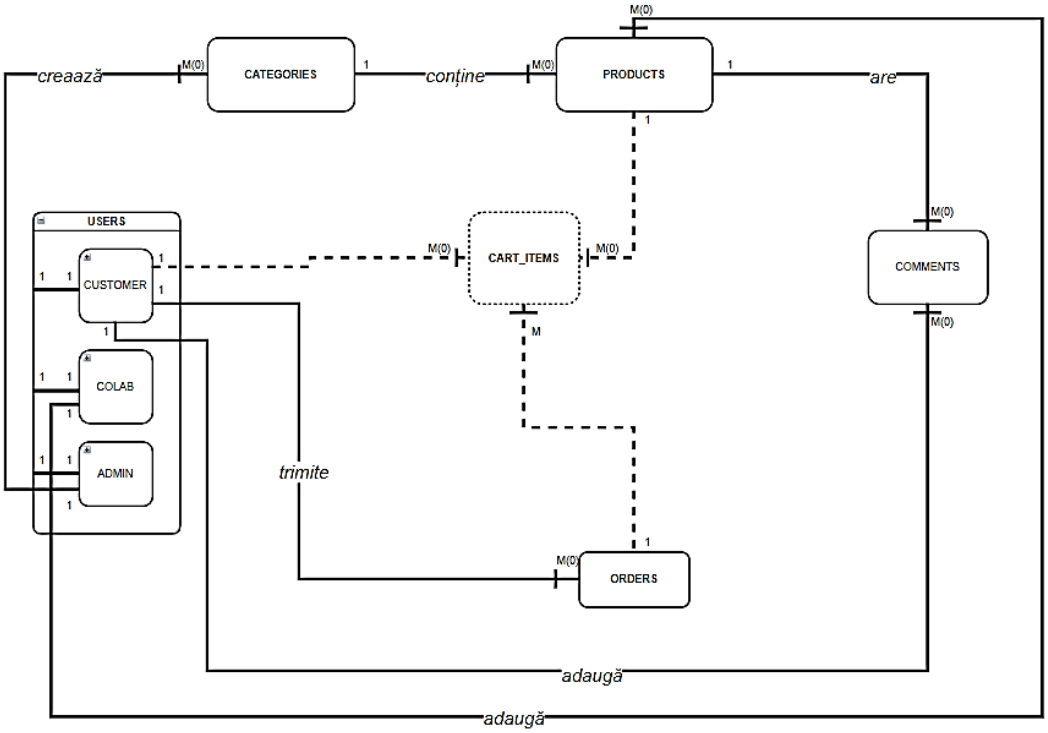
1. Prezentați pe scurt baza de date (utilitatea ei).

Baza de date pe care doresc să o implementez poate fi folosită pentru buna funcționarea a unui magazin online. Baza de date va fi folosită în acest proiect pentru gestionarea unui magazin online de jocuri video (produsele), împărțite în genuri de jocuri(categoriile). Aceasta va avea următoarele tabele:

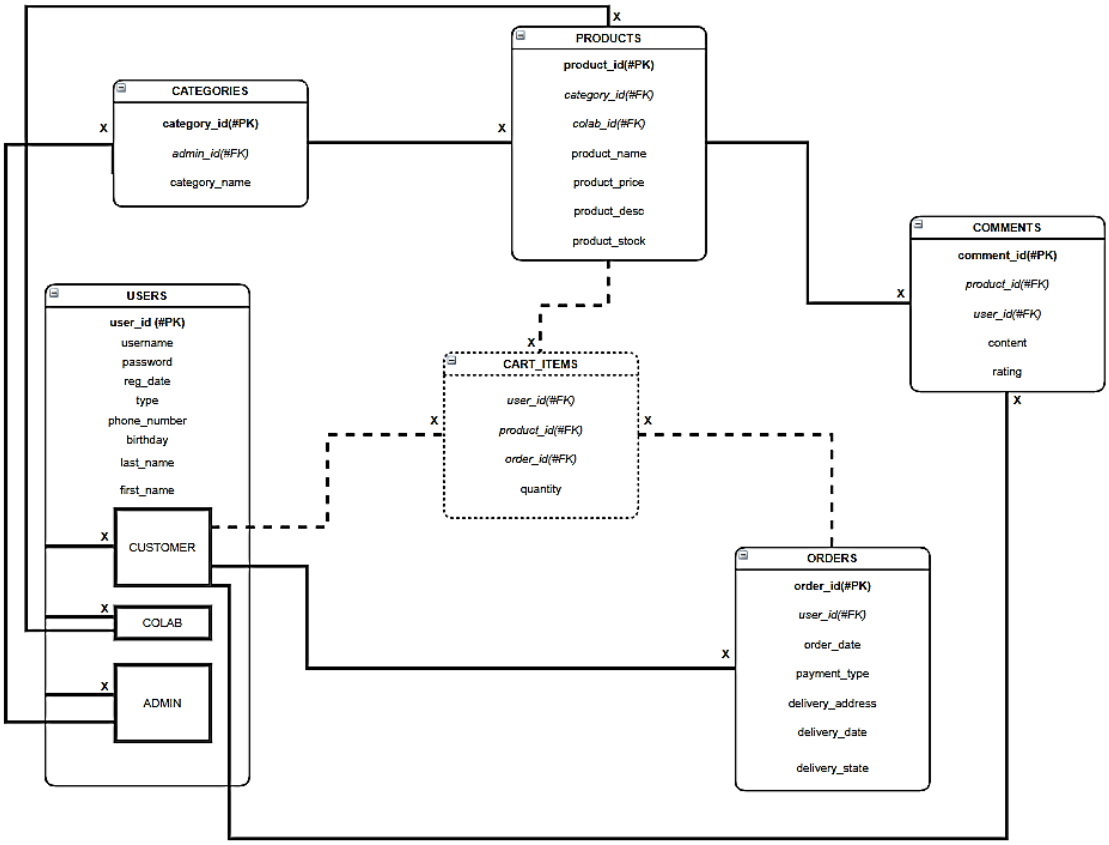
- USERS – reține detaliile fiecărui utilizator (user_id, username, parola, type, reg_date, phone_number, birthday, last_name, first_name). Tipul acestuia care poate fi de 3 feluri: CUSTOMER (client al magazinului online); COLAB (reprezentat al unei companii/firme care dorește să vândă produse prin intermediul site-ului); ADMIN (administrator al site-ului cu drepturi depline);
- CATEGORIES – reține detaliile legate de fiecare categorie(category_id, nume_categorie), dar și id-ul user-ului cine a creat-o, acestea putând fi create/șterse de administratorul site-ului;
- PRODUCTS – reține detaliile legate de fiecare produs în parte(product_id, category_id, colab_id, product_name, product_price, product_stock, product_desc). Colab_Id-ul este id-ul colaboratorului care l-a adăugat;
- COMMENTS – reține detaliile legate de fiecare comentariu lăsat pe site (comment_id, product_id, comment_content);
- CART_ITEMS – este tabela asociativă care are rolul de a reține (user_id, product_id și order_id);
- ORDERS – reține detaliile legate de o comandă (order_id, user_id, order_date, payment_type, delivery_adress, delivery_date, state);

Pentru sistemul de comenzi site-ul va realiza următoarea procesare: Se verifică în tabelul CART_ITEMS produsele al căror ORDER_ID este NULL (adică nu aparțin încă de nici o comandă), iar acelea vor fi produsele care vor aparține de noua comanda. Odata ce sunt atribuite unei comenzi, se modifică ORDER_ID-ul în id-ul respectivei comenzi.

2. Realizați diagrama entitate-relație(ERD).



3. Realizați diagrama conceptuală



4. Implementați în Oracle diagrama conceptuală realizată: definiți toate tabelele, implementând toate constrângerile de integritate necesare(chei primare,cheile externe)

```
CREATE TABLE USERS (
   user_id INT NOT NULL PRIMARY KEY,
   username varchar(30) NOT NULL UNIQUE,
   password varchar(30) NOT NULL,
   reg_date DATE,
   type VARCHAR(30) NOT NULL,
   birthday DATE,
   phone_number VARCHAR(20),
   last_name VARCHAR(20) NOT NULL,
   first_name VARCHAR(20) NOT NULL
);
CREATE TABLE CATEGORIES (
   category_id INT NOT NULL PRIMARY KEY,
   admin_id INT NOT NULL,
   category_name VARCHAR(20) NOT NULL,
   FOREIGN KEY (admin_id) REFERENCES USERS(user_id)
);
CREATE TABLE PRODUCTS (
   product_id INT NOT NULL PRIMARY KEY,
   category_id INT NOT NULL,
   colab_id INT NOT NULL,
   product_name VARCHAR(30) NOT NULL,
   product_price NUMERIC(4,2) NOT NULL,
   product_desc VARCHAR(255),
   product_stock INT,
   FOREIGN KEY (category_id) REFERENCES CATEGORIES(category_id),
   FOREIGN KEY (colab_id) REFERENCES USERS(user_id)
)
```

```
CREATE TABLE COMMENTS(
    comment_id INT NOT NULL PRIMARY KEY,
    user_id INT NOT NULL,
    product_id INT NOT NULL,
    content VARCHAR(255),
    rating INT NOT NULL,
    FOREIGN KEY (user_id) REFERENCES USERS(user_id),
    FOREIGN KEY (product_id) REFERENCES PRODUCTS(product_id)
)
CREATE TABLE ORDERS(
    order_id INT NOT NULL PRIMARY KEY,
    user_id INT NOT NULL,
    order_date DATE NOT NULL,
    payment_type VARCHAR(20) NOT NULL,
    delivery_adress VARCHAR(255),
    delivery_date DATE,
    delivery_state VARCHAR(100),
    FOREIGN KEY (user_id) REFERENCES USERS(user_id)
)
CREATE TABLE CART_ITEMS(
    user_id INT NOT NULL,
    product_id INT NOT NULL,
    order_id INT,
    quantity INT NOT NULL,
    FOREIGN KEY (user_id) REFERENCES USERS(user_id),
    FOREIGN KEY (product_id) REFERENCES PRODUCTS(product_id),
    FOREIGN KEY (order_id) REFERENCES ORDERS(order_id)
)
```

Statement 16

```
CREATE TABLE USERS (
    user_id INT NOT NULL PRIMARY KEY,
    username varchar(30) NOT NULL,
    password varchar(30) NOT NULL,
    reg_date DATE,
    type VARCHAR(30) NOT NULL,
    birthday DATE,
    phone_number VARCHAR(20),
    last_name VARCHAR(20) NOT NULL,
    first_name VARCHAR(20) NOT NULL
)
```

Table created.

Statement 25

```
CREATE TABLE CATEGORIES (
    category_id INT NOT NULL PRIMARY KEY,
    admin_id INT NOT NULL,
    category_name VARCHAR(20) NOT NULL,
    FOREIGN KEY (admin_id) REFERENCES USERS(user_id)
)
```

Table created.

Statement 26

```
CREATE TABLE PRODUCTS (
    product_id INT NOT NULL PRIMARY KEY,
    category_id INT NOT NULL,
    colab_id INT NOT NULL,
    product_name VARCHAR(30) NOT NULL,
    product_price NUMERIC(4,2) NOT NULL,
    product_desc VARCHAR(255),
    product_stock INT,
    FOREIGN KEY (category_id) REFERENCES CATEGORIES(category_id),
    FOREIGN KEY (colab_id) REFERENCES USERS(user_id)
)
```

Table created.

Statement 27

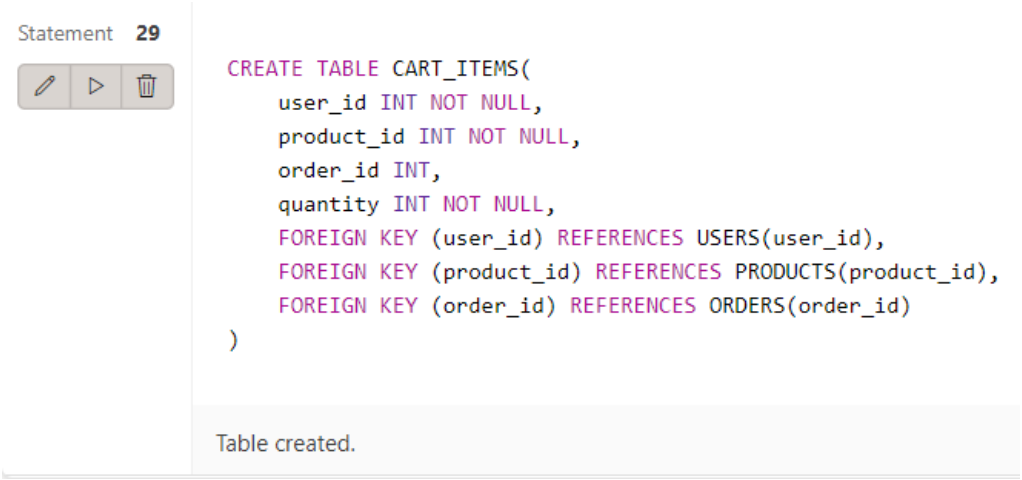
```
CREATE TABLE COMMENTS(
    comment_id INT NOT NULL PRIMARY KEY,
    user_id INT NOT NULL,
    product_id INT NOT NULL,
    content VARCHAR(255),
    rating INT NOT NULL,
    FOREIGN KEY (user_id) REFERENCES USERS(user_id),
    FOREIGN KEY (product_id) REFERENCES PRODUCTS(product_id)
)
```

Table created.

Statement 28

```
CREATE TABLE ORDERS(
    order_id INT NOT NULL PRIMARY KEY,
    user_id INT NOT NULL,
    order_date DATE NOT NULL,
    payment_type VARCHAR(20) NOT NULL,
    delivery_adress VARCHAR(255),
    delivery_date DATE,
    delivery_state VARCHAR(100),
    FOREIGN KEY (user_id) REFERENCES USERS(user_id)
)
```

Table created.



5. Adăugați informații coerente în tabelele create (minim 3-5 înregistrări pentru fiecare entitate independentă; minim 10 înregistrări pentru tabela asociativă).

USERS:

```
INSERT INTO USERS
VALUES
   (1, 'rares_gherasim', 'lorep_ipsum', TO_DATE('2018/07/09', 'yyyy/mm/dd'), 'ADMIN',  TO_DATE('2000/11/04',
'yyyy/mm/dd'), '+40746018999','Gherasim', 'Rares');
INSERT INTO USERS
VALUES
   (2, 'roberta_voinea', 'lorep_ipsum', TO_DATE('2019/03/15', 'yyyy/mm/dd'), 'COLAB',  TO_DATE('2000/12/12',
'yyyy/mm/dd'), '+40753264353','Voinea', 'Roberta');
INSERT INTO USERS
VALUES
   (3, 'matei_corvin', 'lorep_ipsum', TO_DATE('2020/07/15', 'yyyy/mm/dd'), 'CUSTOMER',  TO_DATE('1999/10/06',
'yyyy/mm/dd'), '+40712345678','Corvin', 'Matei');
INSERT INTO USERS
VALUES
   (4, 'sandu_sebastian', 'lorep_ipsum', TO_DATE('2021/01/05', 'yyyy/mm/dd'), 'CUSTOMER',  TO_DATE('2001/05/23',
'yyyy/mm/dd'), '+40787654321','Sandu', 'Sebastian');
INSERT INTO USERS
VALUES
   (5, 'alexandra_topliceanu', 'lorep_ipsum', TO_DATE('2015/09/20', 'yyyy/mm/dd'), 'CUSTOMER',  TO_DATE('2001/01/01',
'yyyy/mm/dd'), '+40787655678','Topliceanu', 'Alexandra');
INSERT INTO USERS
VALUES
   (6, 'moldovan_iulian', 'lorep_ipsum', TO_DATE('2008/06/20', 'yyyy/mm/dd'), 'CUSTOMER',  TO_DATE('2005/10/04',
'yyyy/mm/dd'), '+40712344321','Moldovan', 'Iulian');
```

```
INSERT INTO USERS
VALUES
   (7, 'paun_andrei', 'lorep_ipsum', TO_DATE('1996/06/03', 'yyyy/mm/dd'), 'COLAB',  TO_DATE('2000/12/01', 'yyyy/mm/dd'),
'+40742131234','Paun', 'Andrei');

INSERT INTO USERS
VALUES
(8, 'Florian_gherasim', 'lorep_ipsum', TO_DATE('2018/07/09', 'yyyy/mm/dd'), 'ADMIN',  TO_DATE('2002/10/03',
'yyyy/mm/dd'), '+40746012349','Gherasim', 'Florian');
```

```
116  SELECT * FROM USERS;
117  SELECT * FROM CATEGORIES;
118  SELECT * FROM PRODUCTS;
119  SELECT * FROM COMMENTS;
120  SELECT * FROM ORDERS;
121
122
123
124  INSERT INTO CART_ITEMS
125  VALUES
126
127
128
```

USER_ID	USERNAME	PASSWORD	REG_DATE	TYPE	BIRTHDAY	PHONE_NUMBER	LAST_NAME	FIRST_NAME
8	florin_gherasim	lorep_ipsum	15-MAY-19	ADMIN	23-MAY-02	+4079998014	Gherasim	Florin
7	paun_andrei	lorep_ipsum	03-JUN-96	COLAB	01-DEC-00	+40742131234	Paun	Andrei
1	rares_gherasim	lorep_ipsum	09-JUL-18	ADMIN	04-NOV-00	+40746018999	Gherasim	Rares
2	roberta_voinea	lorep_ipsum	15-MAR-19	COLAB	12-DEC-00	+40753264353	Voinea	Roberta
3	matei_corvin	lorep_ipsum	15-JUL-20	CUSTOMER	06-OCT-99	+40712345678	Corvin	Matei
4	sandu_sebastian	lorep_ipsum	05-JAN-21	CUSTOMER	23-MAY-01	+40787654321	Sandu	Sebastian
5	alexandra_topliceanu	lorep_ipsum	20-SEP-15	CUSTOMER	01-JAN-01	+40787655678	Topliceanu	Alexandra
6	moldovan_iulian	lorep_ipsum	20-JUN-08	CUSTOMER	04-OCT-05	+40712344321	Moldovan	Iulian

CATEGORIES:

```
INSERT INTO CATEGORIES
VALUES
   (1, 1, 'Action');

INSERT INTO CATEGORIES
VALUES
   (2, 1, 'Adventure');

INSERT INTO CATEGORIES
VALUES
   (3, 1, 'RPG');

INSERT INTO CATEGORIES
VALUES
```

(4, 1, 'Shooting');

INSERT INTO CATEGORIES
VALUES
(5, 8, 'VR');

INSERT INTO CATEGORIES
VALUES
(6, 8, 'Logic');

```
116  SELECT * FROM USERS;
117  SELECT * FROM CATEGORIES;
118  SELECT * FROM PRODUCTS;
119  SELECT * FROM COMMENTS;
120  SELECT * FROM ORDERS;
121
122
123
124  INSERT INTO CART_ITEMS
125  VALUES
126
127
128
```

CATEGORY_ID	ADMIN_ID	CATEGORY_NAME
5	8	VR
6	8	Logic
1	1	Action
2	1	Adventure
3	1	RPG
4	1	Shooting

PRODUCTS:

INSERT INTO PRODUCTS
VALUES
(1, 3, 7, 'Minecraft', 29.99, 'Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus erat.', null);

INSERT INTO PRODUCTS
VALUES
(4, 1, 2, 'Fortnite', 49.99, 'Nam ex libero, vehicula non facilisis euismod, suscipit eget odio.', 50);

INSERT INTO PRODUCTS
VALUES
(2, 3, 2, 'GTA V', 99.99, 'Lorem ipsum dolor sit amet, consectetur adipiscing elit', 100);

```
INSERT INTO PRODUCTS
VALUES
   (3, 6, 7, 'Logic', 9.99, 'Fusce eget consequat felis. Sed nec neque eget', 9999);

INSERT INTO PRODUCTS
VALUES
   (5, 2, 7, 'World of Warcraft', 59.99, 'Cras nunc massa, maximus ut ligula et', 10);

INSERT INTO PRODUCTS
VALUES
   (6, 2, 2, 'The Withcer Wild Hunt', 29.99, 'Quisque at laoreet elit, bibendum volutpat lectus.', 30);
```

```
SELECT * FROM USERS;
SELECT * FROM CATEGORIES;
SELECT * FROM PRODUCTS;
SELECT * FROM COMMENTS;
SELECT * FROM ORDERS;



INSERT INTO CART_ITEMS
VALUES
```

PRODUCT_ID	CATEGORY_ID	COLAB_ID	PRODUCT_NAME	PRODUCT_PRICE	PRODUCT_DESC	PRODUCT_STOCK
5	2	7	World of Warcraft	59.99	Cras nunc massa, maximus ut ligula et	10
2	3	2	GTA V	99.99	Lorem ipsum dolor sit amet, consectetur adipiscing elit	100
1	3	7	Minecraft	29.99	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus erat.	-
3	6	7	Logic	9.99	Fusce eget consequat felis. Sed nec neque eget	9999
6	2	2	The Withcer Wild Hunt	29.99	Quisque at laoreet elit, bibendum volutpat lectus.	30
4	1	2	Fortnite	49.99	Nam ex libero, vehicula non facilisis euismod, suscipit eget odio.	50

COMMENTS:

```
INSERT INTO COMMENTS
VALUES
   (1, 2, 6, 'Cel mai fain joc de pe planeta.', 4);

INSERT INTO COMMENTS
VALUES
   (2, 2, 5, 'Nu imi place.', 1);

INSERT INTO COMMENTS
VALUES
   (3, 2, 4, 'Excelent job.', 5);

INSERT INTO COMMENTS
VALUES
   (4, 4, 6, 'Joc zilnic cu prietenii si totul este super.', 5);
```

INSERT INTO COMMENTS
VALUES
(5, 3, 4, 'Nu are logica deloc.', 1);

```
116  SELECT * FROM USERS;
117  SELECT * FROM CATEGORIES;
118  SELECT * FROM PRODUCTS;
119  SELECT * FROM COMMENTS;
120  SELECT * FROM ORDERS;
121
122
123
124  INSERT INTO CART_ITEMS
125  VALUES
126
127
128
```

COMMENT_ID	USER_ID	PRODUCT_ID	CONTENT	RATING
2	2	5	Nu imi place.	1
3	2	4	Excelent job.	5
4	4	6	Joc zilnic cu prietenii si totul este super.	5
5	3	4	Nu are logica deloc.	1
1	2	6	Cel mai fain joc de pe planeta.	4

ORDERS:

INSERT INTO ORDERS
VALUES
(1, 3, TO_DATE('2021/01/01', 'yyyy/mm/dd'), 'CARD', 'Fusce eget consequat felis. Sed nec neque eget', TO_DATE('2021/01/05', 'yyyy/mm/dd'), null);

INSERT INTO ORDERS
VALUES
(2, 4, TO_DATE('2020/11/10', 'yyyy/mm/dd'), 'NUMERAR', 'DSGDGSASDFASDASFAFS. Sed nec neque eget', TO_DATE('2020/11/14', 'yyyy/mm/dd'), null);

INSERT INTO ORDERS
VALUES
(3, 4, TO_DATE('2020/12/25', 'yyyy/mm/dd'), 'NUMERAR', 'DSGDGSASDFASDASFAFS. Sed nec neque eget', TO_DATE('2021/01/05', 'yyyy/mm/dd'), null);

```
120  SELECT * FROM ORDERS;
121
122
123
124  INSERT INTO CART_ITEMS
125  VALUES
126
127
128
```

ORDER_ID	USER_ID	ORDER_DATE	PAYMENT_TYPE	DELIVERY_ADRESS	DELIVERY_DATE	DELIVERY_STATE
1	3	01-JAN-21	CARD	Fusce eget consequat felis. Sed nec neque eget	05-JAN-21	-
2	4	10-NOV-20	NUMERAR	DSGDGSASDFASDASFAFS. Sed nec neque eget	14-NOV-20	-
3	4	25-DEC-20	NUMERAR	DSGDGSASDFASDASFAFS. Sed nec neque eget	05-JAN-21	-

CART_ITEMS:

```
INSERT INTO CART_ITEMS
VALUES
   (4, 5, 2, 3);

INSERT INTO CART_ITEMS
VALUES
   (4, 2, 2, 4);

INSERT INTO CART_ITEMS
VALUES
   (3, 2, 1, 5);

INSERT INTO CART_ITEMS
VALUES
   (4, 3, 2, 1);

INSERT INTO CART_ITEMS
VALUES
   (3, 5, 1, 2);

INSERT INTO CART_ITEMS
VALUES
   (4, 6, 2, 10);

INSERT INTO CART_ITEMS
VALUES
   (4, 2, 2, 7);

INSERT INTO CART_ITEMS
VALUES
   (4, 5, 3, 6);

INSERT INTO CART_ITEMS
VALUES
   (3, 3, 1, 5);

INSERT INTO CART_ITEMS
VALUES
   (3, 6, 1, 4);
```

```
INSERT INTO CART_ITEMS
VALUES
  (3, 4, 1, 3);

INSERT INTO CART_ITEMS
VALUES
  (4, 6, 3, 2);

INSERT INTO CART_ITEMS
VALUES
  (4, 3, 3, 12);

INSERT INTO CART_ITEMS
VALUES
  (3, 1, null, 1);

INSERT INTO CART_ITEMS
VALUES
  (5, 3, null, 3);

INSERT INTO CART_ITEMS
VALUES
  (5, 2, null, 10);
```

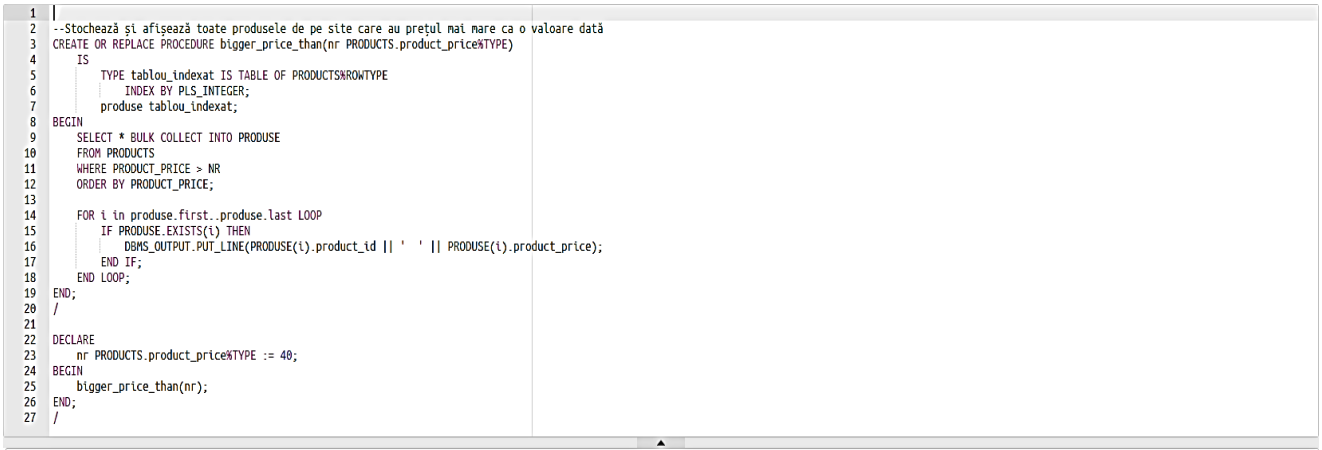
```
181  SELECT * FROM CART_ITEMS;
```

USER_ID	PRODUCT_ID	ORDER_ID	QUANTITY
4	5	2	3
4	2	2	4
3	2	1	5
4	3	2	1
3	5	1	2
4	6	2	10
4	2	2	7
4	5	3	6
3	3	1	5
3	6	1	4
3	4	1	3
4	6	3	2
4	3	3	12
3	1	-	1
5	3	-	3
5	2	-	10

6. Definiți un subprogram stocat care să utilizeze un tip de colecție studiat. Apelați subprogramul.

--Stochează și afișează toate produsele de pe site care au prețul mai mare ca o valoare dată

```
CREATE OR REPLACE PROCEDURE bigger_price_than(nr PRODUCTS.product_price%TYPE)
  IS
    TYPE tablou_indexat IS TABLE OF PRODUCTS%ROWTYPE
      INDEX BY PLS_INTEGER;
    produse tablou_indexat;
BEGIN
  SELECT * BULK COLLECT INTO PRODUSE
  FROM PRODUCTS
  WHERE PRODUCT_PRICE > NR
  ORDER BY PRODUCT_PRICE;
  FOR i in produse.first..produse.last LOOP
    IF PRODUSE.EXISTS(i) THEN
      DBMS_OUTPUT.PUT_LINE(PRODUSE(i).product_id || ' ' || PRODUSE(i).product_price);
    END IF;
  END LOOP;
END;
/
DECLARE
  nr PRODUCTS.product_price%TYPE := 40;
BEGIN
  bigger_price_than(nr);
END;
/
```



```
--Stochează și afișează toate produsele de pe site care au prețul mai mare ca o valoare dată
CREATE OR REPLACE PROCEDURE bigger_price_than(nr PRODUCTS.product_price%TYPE)
    IS
        TYPE tablou_indexat IS TABLE OF PRODUCTS%ROWTYPE
            INDEX BY PLS_INTEGER;
        produse tablou_indexat;
BEGIN
    SELECT * BULK COLLECT INTO PRODUSE
    FROM PRODUCTS
    WHERE PRODUCT_PRICE > NR
    ORDER BY PRODUCT_PRICE;

    FOR i in produse.first..produse.last LOOP
        IF PRODUSE.EXISTS(i) THEN
            DBMS_OUTPUT.PUT_LINE(PRODUSE(i).product_id || ' ' || PRODUSE(i).product_price);
        END IF;
    END LOOP;
END;
/

DECLARE
    nr PRODUCTS.product_price%TYPE := 40;
BEGIN
    bigger_price_than(nr);
END;
/
```

```
Procedure created.


Statement processed.
4  49.99
5  59.99
2  99.99
```

7. Definiți un subprogram stocat care să utilizeze un tip de cursor studiat. Apelați subprogramul.

--Obtineți pentru fiecare categorie numele acesteia și numarul produselor ce apartin de aceasta utilizand un cursor explicit.

```
CREATE OR REPLACE PROCEDURE CATEG_AND_NR
IS
   CURSOR C IS
      SELECT category_name nume, COUNT(product_id) nr
      FROM CATEGORIES c, PRODUCTS p
      WHERE c.category_id = p.category_id(+)
      GROUP BY category_name
      ORDER BY category_name;
   var_name categories.category_name%type;
   var_nr int;
BEGIN
   OPEN C;
   LOOP
      FETCH C INTO VAR_NAME, VAR_NR;
      EXIT WHEN C%NOTFOUND;
      IF (VAR_NR = 0) THEN
         DBMS_OUTPUT.PUT_LINE('Categoria ' || VAR_NAME || ' nu conține produse.');
      ELSIF (VAR_NR = 1) THEN
         DBMS_OUTPUT.PUT_LINE('Categoria ' || VAR_NAME || ' conține un singur produs.');
      ELSE
         DBMS_OUTPUT.PUT_LINE('Categoria ' || VAR_NAME || ' conține ' || VAR_NR || ' produse.');
      END IF;
   END LOOP;
END;
```

```
1  --Obtineți pentru fiecare categorie numele acesteia și numarul produselor ce apartin de aceasta utilizand un cursor explicit.
2  CREATE OR REPLACE PROCEDURE CATEG_AND_NR
3  IS
4      CURSOR C IS
5          SELECT category_name nume, COUNT(product_id) nr
6          FROM CATEGORIES c, PRODUCTS p
7          WHERE c.category_id = p.category_id(+)
8          GROUP BY category_name
9          ORDER BY category_name;
10     var_name categories.category_name%type;
11     var_nr int;
12 BEGIN
13     OPEN C;
14     LOOP
15         FETCH C INTO VAR_NAME, VAR_NR;
16         EXIT WHEN C%NOTFOUND;
17         IF (VAR_NR = 0) THEN
18             DBMS_OUTPUT.PUT_LINE('Categoria ' || VAR_NAME || ' nu conține produse.');
19         ELSIF (VAR_NR = 1) THEN
20             DBMS_OUTPUT.PUT_LINE('Categoria ' || VAR_NAME || ' conține un singur produs.');
21         ELSE
22             DBMS_OUTPUT.PUT_LINE('Categoria ' || VAR_NAME || ' conține ' || VAR_NR || ' produse.');
23         END IF;
24     END LOOP;
25 END;
26
27 BEGIN
28     CATEG_AND_NR;
29 END;
```

```
Statement processed.
Categoria Action conține un singur produs.
Categoria Adventure conține 2 produse.
Categoria Logic conține un singur produs.
Categoria RPG conține 2 produse.
Categoria Shooting nu conține produse.
Categoria VR nu conține produse.
```

8. Definiți un subprogram stocat de tip funcție care să utilizeze 3 dintre tabelele definite. Tratați toate excepțiile care pot apărea. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.

--Se cere să se afle numarul de comentarii maxim pentru cel mai comentat produs dintr-o anumita categorie, data prin nume.

```
CREATE OR REPLACE FUNCTION CATEG_PROD_COMM(nume_cat categories.category_name%type)
RETURN INTEGER IS
   TYPE vector IS VARRAY(20) OF NUMBER(4);
   id_produse vector;
   numar NUMBER(4);
   aux NUMBER(4);
   id_categorie categories.category_id%type;
   exista_categoria EXCEPTION;
   mai_multe_cu_acelasi_nume EXCEPTION;
   are_produse EXCEPTION;
   PROD_NU_AU_COMM EXCEPTION;
BEGIN
   numar := 0;
   SELECT COUNT(CATEGORY_NAME) into numar
   FROM CATEGORIES
   WHERE CATEGORY_NAME = NUME_CAT;

   IF (numar = 0) THEN RAISE exista_categoria;
   ELSIF (numar > 1) THEN RAISE mai_multe_cu_acelasi_nume;
   end if;
   numar := 0;

   SELECT CATEGORY_ID into id_categorie
   FROM CATEGORIES
   WHERE CATEGORY_NAME = nume_cat;

   SELECT count(*) INTO NUMAR
   FROM PRODUCTS
   WHERE CATEGORY_ID = ID_CATEGORIE;

   IF (numar = 0) THEN RAISE are_produse;
   END IF;
```

```
    SELECT PRODUCT_ID BULK COLLECT INTO id_produse
    FROM PRODUCTS
    WHERE CATEGORY_ID = id_categorie;


    numar := 0; --retin maximul in el
    FOR i in id_produse.first..id_produse.last LOOP
        aux := 0;
        SELECT COUNT(*) into AUX
        FROM COMMENTS
        WHERE PRODUCT_ID = id_produse(i);
        IF (AUX > NUMAR) THEN
            NUMAR := AUX;
        END IF;
    END LOOP;

    IF (NUMAR = 0) THEN RAISE PROD_NU_AU_COMM;
    END IF;

    RETURN NUMAR;

    EXCEPTION
        WHEN exista_categoria THEN
            DBMS_OUTPUT.PUT_LINE('Nu exista categoria cautata.');
            RETURN -1;
        WHEN mai_multe_cu_acelasi_nume THEN
            DBMS_OUTPUT.PUT_LINE('Exista mai multe categorii cu acest nume.');
            RETURN -1;
        WHEN are_produse THEN
            DBMS_OUTPUT.PUT_LINE('Categoria cerută nu are produse.');
            RETURN -1;
        WHEN PROD_NU_AU_COMM THEN
            DBMS_OUTPUT.PUT_LINE('Niciun produs din categoria ceruta nu are comentarii.');
            RETURN -1;
END;
/
```

-------*Output-uri corecte pentru funcție:*

```
DECLARE
    text categories.category_name%type;
BEGIN
    text := 'Action';
    DBMS_OUTPUT.PUT_LINE(CATEG_PROD_COMM(text));
END;
```

```
Statement processed.
2
```

```
DECLARE
    text categories.category_name%type;
BEGIN
    text := 'Adventure';
    DBMS_OUTPUT.PUT_LINE(CATEG_PROD_COMM(text));
END;
```

```
Statement processed.
2
```

------*Cazul în care nu exista categoria cautata:*

```
DECLARE
    text categories.category_name%type;
BEGIN
    text := 'salam';
    DBMS_OUTPUT.PUT_LINE(CATEG_PROD_COMM(text));
END;
```

```
Statement processed.
Nu exista categoria cautata.
-1
```

--------*Cazul în care în categoria cerută nu există produse:*

```
DECLARE
    text categories.category_name%type;
BEGIN
    text := 'Shooting'; --categoria asta nu are jocuri asociate
    DBMS_OUTPUT.PUT_LINE(CATEG_PROD_COMM(text));
END;
```

```
Statement processed.
Categoria cerută nu are produse.
-1
```

---------*Cazul în care nici un produs din categoria ceruta nu are comentarii:*

```
DECLARE
    text categories.category_name%type;
BEGIN
    text := 'Logic';
    DBMS_OUTPUT.PUT_LINE(CATEG_PROD_COMM(text));
END;
```

```
Statement processed.
Niciun produs din categoria ceruta nu are comentarii.
-1
```

9. Definiți un subprogram stocat de tip procedură care să utilizeze 5 dintre tabelele definite. Tratați toate excepțiile care pot apărea. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.

--Sa se determine cate produse din fiecare categorie a cumparat un customer. Pentru a fi deja cumparate, stare livrarii din order trebuie sa fie 'Done'
--Stiind doar nume, prenumele și nr-ul de telefon al acestuia.

```
CREATE OR REPLACE PROCEDURE EX9(nume USERS.last_name%type , prenume USERS.first_name%type,
telefon USERS.phone_number%type)
IS
   TYPE tablou_indexat IS TABLE OF CART_ITEMS%ROWTYPE
                  INDEX BY PLS_INTEGER;
   TYPE tablou_indexat2 is TABLE OF CATEGORIES%ROWTYPE
                  INDEX BY PLS_INTEGER;
   id_uri tablou_indexat;
   categ tablou_indexat2;
   TYPE vector IS VARRAY(40) OF CATEGORIES.CATEGORY_ID%TYPE;
   categ_tabel vector;
   nume_categ CATEGORIES.CATEGORY_NAME%TYPE;
   aux number(4);
   id_utilizator USERS.user_id%type := '-1';
   nu_are_comenzi_finalizate EXCEPTION;
BEGIN
   aux := 0;

   SELECT user_id into id_utilizator
   FROM USERS
   WHERE last_name = nume AND first_name = prenume and PHONE_NUMBER = telefon;

   SELECT CAR.USER_ID, CAR.PRODUCT_ID, CAR.ORDER_ID, CAR.QUANTITY BULK COLLECT INTO
ID_URI
   FROM CART_ITEMS CAR, ORDERS ORD
   WHERE CAR.USER_ID = ID_UTILIZATOR AND NVL(CAR.ORDER_ID, 0) = ORD.ORDER_ID
      AND ORD.DELIVERY_STATE ='Done'; -- ORDER_ID IS NOT NULL ==> COMANDA TRIMISA.

   IF (id_uri.count = 0) THEN RAISE nu_are_comenzi_finalizate;
   END IF;

   SELECT * BULK COLLECT INTO CATEG
   FROM CATEGORIES;

   CATEG_TABEL := VECTOR();
   FOR I IN ID_URI.FIRST..ID_URI.LAST LOOP
      SELECT CATEGORY_ID INTO NUME_CATEG
      FROM PRODUCTS
      WHERE ID_URI(I).PRODUCT_ID = PRODUCT_ID;
      CATEG_TABEL.EXTEND();
      CATEG_TABEL(i) := NUME_CATEG;
   END LOOP;

   FOR I IN CATEG.FIRST..CATEG.LAST LOOP
      aux := 0;
      FOR J IN CATEG_TABEL.FIRST..CATEG_TABEL.LAST LOOP
```

```
        IF (CATEG(I).CATEGORY_ID = CATEG_TABEL(J)) THEN
          aux := aux + 1;
        END IF;
      END LOOP;

      IF (AUX != 0) THEN
        DBMS_OUTPUT.PUT(CATEG(I).CATEGORY_NAME || ' ');
        AUX := 0;
        FOR J IN CATEG_TABEL.FIRST..CATEG_TABEL.LAST LOOP
          IF (CATEG(I).CATEGORY_ID = CATEG_TABEL(J)) THEN
            AUX := AUX + ID_URI(J).QUANTITY;
          END IF;
        END LOOP;
        DBMS_OUTPUT.PUT(AUX);
        DBMS_OUTPUT.PUT_LINE('');
      END IF;
    END LOOP;

    EXCEPTION
      WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Nu exista utilizator cu datele introduse.');
      WHEN nu_are_comenzi_finalizate THEN
        DBMS_OUTPUT.PUT_LINE('Utilizatorul nu are comenzi finalizate.');
END;
```

```
70  BEGIN
71      EX9('Corvin', 'Matei', '+40712345678');
72  END;
```

```
Statement processed.
Action 3
Adventure 6
RPG 5
Logic 5
```

```
70  BEGIN
71      EX9('Sandu', 'Sebastian', '+40787654321');
72  END;
73
74  SELECT * FROM USERS;
```

```
Statement processed.
Adventure 13
RPG 11
Logic 1
```

Atunci cand utilizatorul nu exista:

```
71  BEGIN
72      EX9('Gherasim', 'Raresc', '+40746018999');
73  END;
```

```
Statement processed.
Nu exista utilizator cu datele introduse.
```

Atunci cand utilizatorul nu are comenzi finalizate:

```
72  BEGIN
73      EX9('Gherasim', 'Rares', '+40746018999');
74  END;
75
76
```

```
Statement processed.
Utilizatorul nu are comenzi finalizate.
```

10. Definiți un *trigger* de tip LMD la nivel de comandă. Declanșați *trigger*-ul.

Trigger-ul are rolul de a nu lăsa să se facă actualizări decât de luni până vineri între orele 8 si 17 asupra tabelului PRODUCTS.

```
CREATE OR REPLACE TRIGGER trig1
   BEFORE INSERT OR UPDATE OR DELETE ON PRODUCTS
BEGIN
   IF (TO_CHAR(SYSDATE,'D') = 1 OR TO_CHAR(SYSDATE,'D') = 2) OR
(TO_CHAR(SYSDATE,'HH24') NOT BETWEEN 8 AND 17)
   THEN
      RAISE_APPLICATION_ERROR(-20001,'tabelul nu poate fi actualizat');
   END IF;
END;
/
```

Insert oprit de trigger:

```
11  INSERT INTO PRODUCTS
12  VALUES (7,3,2,'Salam', 30.11, 'Lorem ipsum dolor sit amet, consectetur adipiscing elit', 1);
13
14  SELECT * FROM PRODUCTS;
```

```
ORA-20001: tabelul nu poate fi actualizat ORA-06512: at "SQL_LQAEINXFCFRGIFYSEIVZIEROM.TRIG1", line 4
ORA-06512: at "SYS.DBMS_SQL", line 1721
```

11. Definiți un *trigger* de tip LMD la nivel de linie. Declanșați *trigger*-ul.
Trigger-ul are rolul de a nu lăsa să fie scăzut sub valoarea de 10 euro prețul unui joc.

```
CREATE OR REPLACE TRIGGER trig2
   BEFORE UPDATE OF product_price ON PRODUCTS
   FOR EACH ROW
BEGIN
   IF (:NEW.product_price < 10) THEN
      RAISE_APPLICATION_ERROR(-20002,'Pretul unui produs nu poate sa fie mai mic decat
10 euro.');
   END IF;
END;
/
```

```
11  UPDATE Products
12  SET PRODUCT_PRICE = PRODUCT_PRICE - 10;
13
```

```
ORA-20002: Pretul unui produs nu poate sa fie mai mic decat 10 euro. ORA-06512: at "SQL_LQAEINXFCFRGIFYSEIVZIEROM.TRIG2", line 3
ORA-06512: at "SYS.DBMS_SQL", line 1721
```

12. Definiți un *trigger* de tip LDD. Declanșați *trigger*-ul.

Trigger-ul scrie in tabelul SYSTEM_UPDATES de feicare data când are loc o schimbare asupra bazei de date.

```
CREATE TABLE system_updates
   (event VARCHAR(20),
    name_obj VARCHAR(30),
    data DATE);


CREATE OR REPLACE TRIGGER trig3
   BEFORE CREATE OR DROP OR ALTER ON SCHEMA
BEGIN
   INSERT INTO system_updates
      VALUES (  SYS.SYSEVENT, SYS.DICTIONARY_OBJ_NAME, SYSDATE);
END;
/
```

```
16  CREATE TABLE SALAM (
17      id_salam int,
18      nume_salam VARCHAR(20)
19      )
20
21  ALTER TABLE SALAM
22  DROP COLUMN nume_salam;
23
24  DROP TABLE SALAM;
25
26  select * from system_updates;
27
28
```

EVENT	NAME_OBJ	DATA
CREATE	SALAM	09-JAN-21
ALTER	SALAM	09-JAN-21
DROP	SALAM	09-JAN-21