Cozma Laura-Elena
BDTS, grupa 405

Proiect SBD

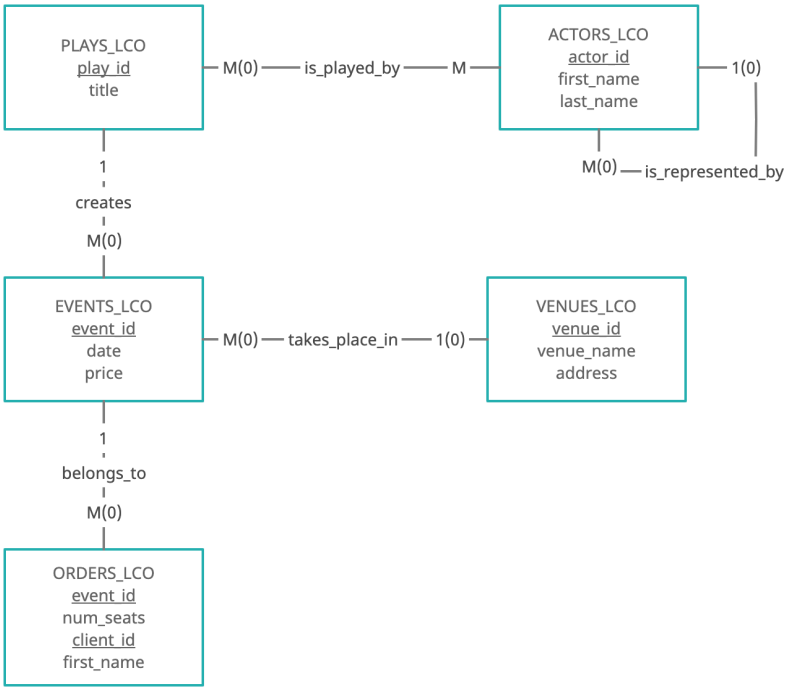
a. Prezentarea bazei de date

Baza de date aleasă poate fi folosită în cadrul unui sistem de vânzare de bilete de teatru. Aceasta se bazează pe existența mai multor piese de teatru. Între acestea și actori există o relație de tipul many-to-many, deoarece într-o piesă pot juca mai mulți actori, iar un actor joacă în mai multe piese.

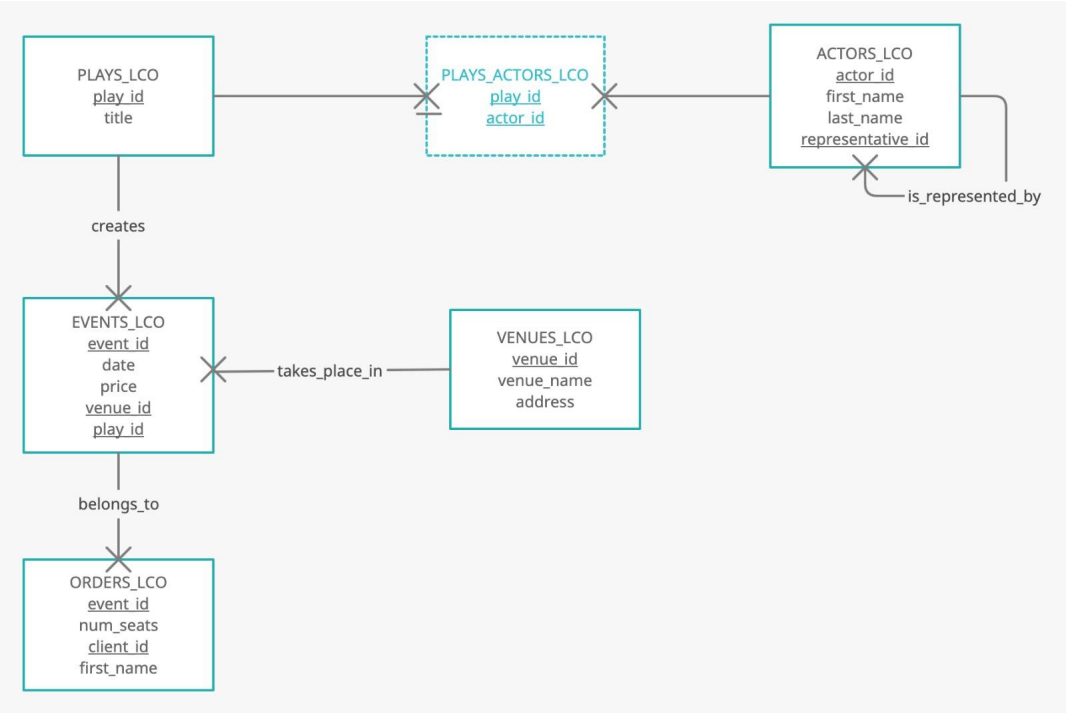
De asemenea, un aspect important îl reprezintă existența evenimentelor. Fiecarei piese de teatru îi sunt asociate unul sau mai multe evenimente, care conțin data, ora și prețul spectacolului. Fiecare eveniment are asociată și o locație (sau aceasta poate să fie inexistentă în cazul în care nu se cunoaște încă locația). Aceasta are drept atribute numele sălii și capacitatea acesteia, dar și id-ul adresei instituției în care se află. O instituție poate avea mai multe săli (de exemplu Teatrul Național București are Sala Pictura și Sala Studio).

În cadrul sistemului un client poate să își achiziționeze un număr de bilete pentru spectacolul ales. Fiecare comanda(order) va conține o referință către client, dar și către evenimentul ales și numărul de bilete cumpărate de un client.

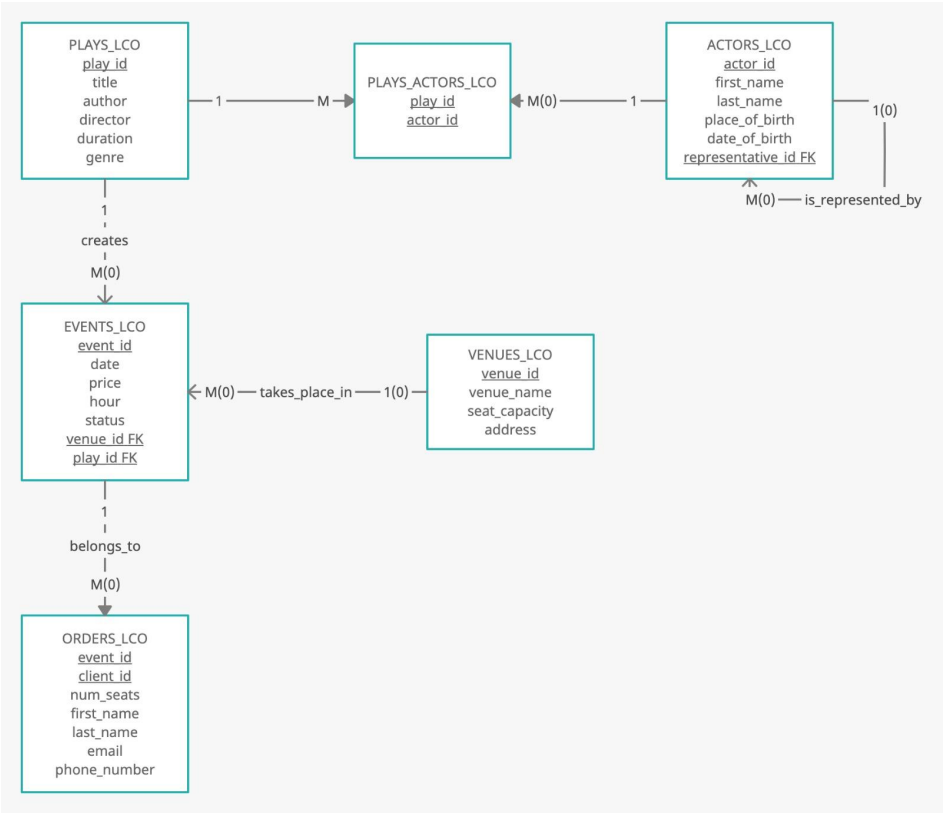
b. Realizarea diagramei entitate-relație (ERD)



c. Realizarea diagramei conceptuale



d. Transformarea sistemului conceptual într-un design logic, subliniind relaţiile dintre tabele, cheile primare şi străine (externe).



e. Transformarea design-ului logic în design fizic, astfel încât tabela să fie în FN3.

- Exemplu de atribut multivaloare

Tabelul venues_lco are urmatoarea structură:
VENUES_LCO(#venue_id, venue_name, seat_capacity, address)
Address este un atribut multivaloare, el având următoarea formă:

venue_id	venue_name	seat_capacity	address
1	Sala Pictura	100	Teatrul Odeon, Bucuresti
2	Sala Studio	85	Teatrul Odeon, Bucuresti
3	Sala Mare	300	Teatrul Nottara, Bucuresti

Conform algoritmului FN1, se înlocuiesc în tabel coloanele corespunzătoare atributelor compuse cu coloane ce conţin componentele elementare ale acestora. Astfel, în venue_lco coloana address va fi inlocuita cu institution_name și city.

VENUES_LCO(#venue_id, venue_name, seat_capacity, institution_name, city)

venue_id	venue_name	seat_capacity	institution_name	city
1	Sala Pictura	100	Teatrul Odeon	Bucuresti
2	Sala Studio	85	Teatrul Odeon	Bucuresti
3	Sala Mare	300	Teatrul Nottara	Bucuresti

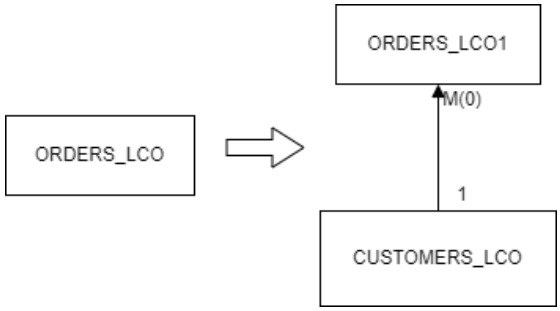
Nu mai exista alte atribute multivaloare.

- Exemplu de tabel relațional din diagramă care e în FN1, dar nu e în FN2

ORDERS_LCO(#event_id, #client_id, num_seats, first_name, last_name, email, phone_number)
Tabelul orders_lco este în FN1, dar nu este în FN2, deoarece first_name, last_name, email și phone_number depind parțial de cheia primară.
Se fac următoarele transformări:

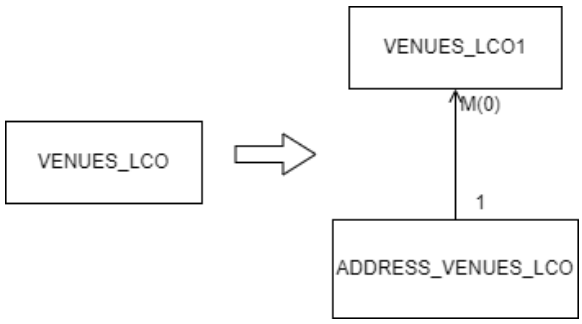
- Se creează un nou tabel customers_lco ce va conține client_id, first_name, last_name, phone_number

- Din orders_lco se creează orders_lco1 care va conține numai event_id, client_id și num_seats
  CUSTOMERS_LCO(#customer_id, first_name, last_name, phone_number, email)
  ORDERS_LCO1(#event_id, #customer_id, num_seats)

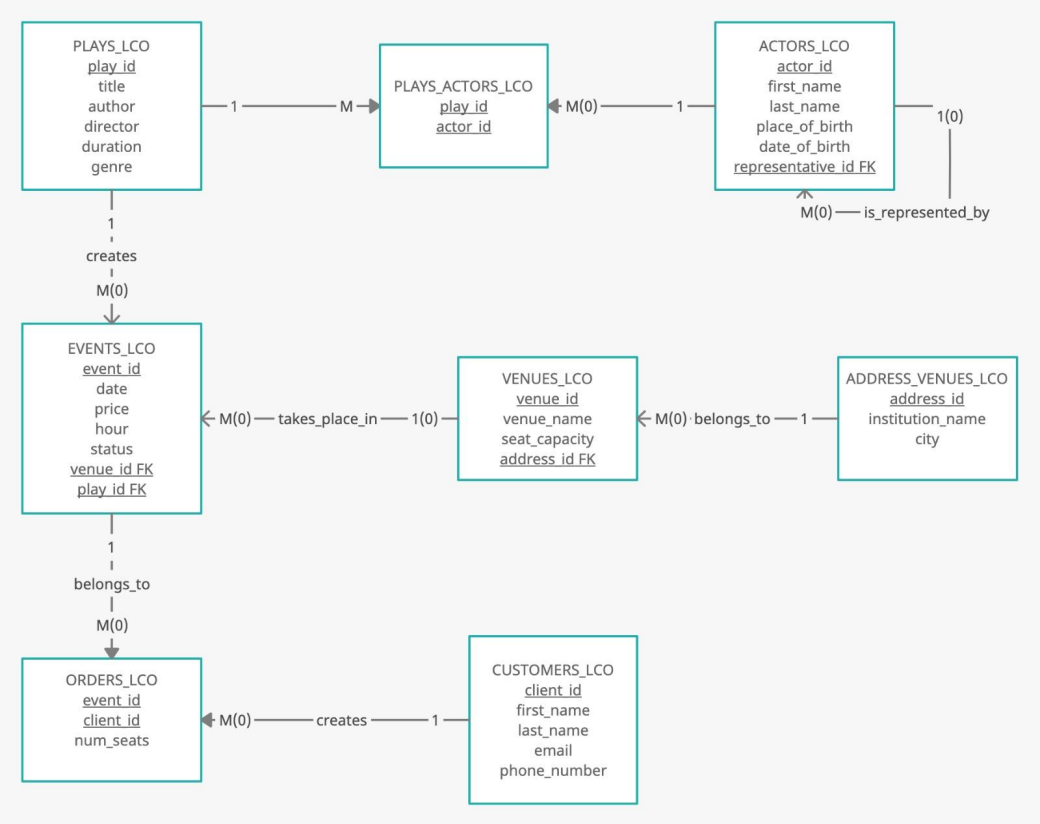


- Exemplu de tabel relațional din diagramă care e în FN2, dar nu e în FN3
  Tabelul venues_lco nu este în continuare în FN3. Vedem ca perechea Teatrul Odeon - Bucuresti se repetă, astfel existând o dependență tranzitivă. Formăm noile tabele:

  VENUES_LCO1(#venue_id, venue_name, seat_capacity, address_id)
  ADDRESS_VENUES_LCO(#address_id, institution_name, city)



  Design-ul fizic este:
  ACTORS_LCO(#actor_id, first_name, last_name, date_of_birth, place_of_birth, representative_id FK)
  PLAYS_LCO(#play_id, title, author, director, duration, genre)
  PLAYS_ACTORS_LCO(#play_id, #actor_id)
  EVENTS_LCO(#event_id, date, price, hour, status, play_id, venue_id)
  VENUES_LCO(#venue_id, venue_name, seat_capacity, address_id)
  ADDRESS_VENUES_LCO(#address_id, institution_name, city)
  CUSTOMERS_LCO(#customer_id, first_name, last_name, phone_number, email)
  ORDERS_LCO(#event_id, #customer_id, num_seats)



f. Implementarea tabelelor în Oracle
Actors_lco

```
CREATE TABLE actors_lco
    (actor_id NUMBER(5) PRIMARY KEY,
     first_name VARCHAR2(20) NOT NULL,
     last_name VARCHAR2(20) NOT NULL,
     place_of_birth VARCHAR2(20),
     date_of_birth DATE,
     representative_id NUMBER(5) REFERENCES actors_lco(actor_id));

CREATE SEQUENCE actors_lco_seq
START WITH 1
INCREMENT BY 1;

INSERT INTO actors_lco (actor_id, first_name, last_name, place_of_birth,
date_of_birth)
VALUES (actors_lco_seq.NEXTVAL, 'Victor', 'Rebengiuc', 'Bucuresti',
TO_DATE('10/02/1933', 'DD/MM/YYYY'));
INSERT INTO actors_lco (actor_id, first_name, last_name, place_of_birth,
```

```
date_of_birth, representative_id)
VALUES (actors_lco_seq.NEXTVAL, 'Oana', 'Pellea', 'Bucuresti',
TO_DATE('29/01/1962', 'DD/MM/YYYY'), 1);
INSERT INTO actors_lco (actor_id, first_name, last_name, place_of_birth,
date_of_birth)
VALUES (actors_lco_seq.NEXTVAL, 'Maia', 'Morgenstern', 'Bucuresti',
TO_DATE('01/05/1962', 'DD/MM/YYYY'));
INSERT INTO actors_lco (actor_id, first_name, last_name, place_of_birth,
date_of_birth, representative_id)
VALUES (actors_lco_seq.NEXTVAL, 'Claudiu', 'Bleont', 'Bucuresti',
TO_DATE('17/08/1959', 'DD/MM/YYYY'), 1);
INSERT INTO actors_lco (actor_id, first_name, last_name, place_of_birth,
date_of_birth, representative_id)
VALUES (actors_lco_seq.NEXTVAL, 'Alexandru', 'Arsinel', 'Dolhasca',
TO_DATE('04/06/1939', 'DD/MM/YYYY'), 3);

SELECT * FROM actors_lco;
```

Plays_lco

```
CREATE TABLE plays_lco
    (play_id NUMBER(5) PRIMARY KEY,
     title VARCHAR2(50) NOT NULL,
     author VARCHAR2(20),
     director VARCHAR2(20),
     duration NUMBER(3) NOT NULL,
     genre VARCHAR2(20) NOT NULL);

CREATE SEQUENCE plays_lco_seq
START WITH 1
INCREMENT BY 1;

INSERT INTO plays_lco (play_id, title, author, director, duration, genre)
VALUES (plays_lco_seq.NEXTVAL, 'Padurea spanzuratilor', 'Liviu Rebreanu',
'Radu Afrim', 190, 'drama');
INSERT INTO plays_lco (play_id, title, author, director, duration, genre)
VALUES (plays_lco_seq.NEXTVAL, 'Trei surori', 'Anton Cehov', 'Radu Afrim',
180, 'drama');
INSERT INTO plays_lco (play_id, title, author, director, duration, genre)
VALUES (plays_lco_seq.NEXTVAL, 'Doua loturi', 'I.L. Caragiale', 'Alexandru
```

```
Dabija', 60, 'comedie');
INSERT INTO plays_lco (play_id, title, author, director, duration, genre)
VALUES (plays_lco_seq.NEXTVAL, 'Toti fiii mei', 'Arthur Miller', 'Ion
Caramitru', 160, 'drama');
INSERT INTO plays_lco (play_id, title, director, duration, genre)
VALUES (plays_lco_seq.NEXTVAL, 'Jaful', 'Vlad Massaci', 130, 'drama');
SELECT * FROM plays_lco;
```

Plays_actors_lco

```
CREATE TABLE plays_actors_lco
    (play_id NUMBER(5) REFERENCES plays_lco(play_id) ON DELETE CASCADE,
     actor_id NUMBER(5) REFERENCES actors_lco(actor_id) ON DELETE CASCADE,
     CONSTRAINT plays_actors_lco_pk PRIMARY KEY(play_id, actor_id));

INSERT INTO plays_actors_lco (play_id, actor_id)
VALUES(1, 1);
INSERT INTO plays_actors_lco (play_id, actor_id)
VALUES(1, 3);
INSERT INTO plays_actors_lco (play_id, actor_id)
VALUES(1, 4);
INSERT INTO plays_actors_lco (play_id, actor_id)
VALUES(2, 2);
INSERT INTO plays_actors_lco (play_id, actor_id)
VALUES(2, 3);
INSERT INTO plays_actors_lco (play_id, actor_id)
VALUES(2, 5);
INSERT INTO plays_actors_lco (play_id, actor_id)
VALUES(4, 1);
INSERT INTO plays_actors_lco (play_id, actor_id)
VALUES(4, 4);
INSERT INTO plays_actors_lco (play_id, actor_id)
VALUES(4, 5);
INSERT INTO plays_actors_lco (play_id, actor_id)
VALUES(3, 1);


SELECT * FROM plays_actors_lco;
```

Events_lco

```
CREATE TABLE events_lco
    (event_id NUMBER(5) PRIMARY KEY,
     play_id NUMBER(5) REFERENCES plays_lco(play_id) ON DELETE CASCADE,
     venue_id NUMBER(5) REFERENCES venues_lco(venue_id) ON DELETE CASCADE,
     event_date DATE NOT NULL,
     price NUMBER(4, 1) NOT NULL,
     hour VARCHAR2(5) NOT NULL,
     status VARCHAR2(20));

CREATE SEQUENCE events_lco_seq
START WITH 1
INCREMENT BY 1;

INSERT INTO events_lco(event_id, play_id, venue_id, event_date, price,
hour, status)
VALUES(events_lco_seq.NEXTVAL, 1, 1, TO_DATE('30/01/2022', 'DD/MM/YYYY'),
60, '20:00', 'ACTIV');
INSERT INTO events_lco
VALUES(events_lco_seq.NEXTVAL, 1, 1, TO_DATE('15/02/2022', 'DD/MM/YYYY'),
60, '20:00', 'ANULAT');
INSERT INTO events_lco
VALUES(events_lco_seq.NEXTVAL, 4, 2, TO_DATE('04/02/2022', 'DD/MM/YYYY'),
90, '19:30', 'ACTIV');
INSERT INTO events_lco(event_id, play_id, venue_id, event_date, price,
hour)
VALUES(events_lco_seq.NEXTVAL, 4, 2, TO_DATE('04/12/2021', 'DD/MM/YYYY'),
80, '20:30');
INSERT INTO events_lco
VALUES(events_lco_seq.NEXTVAL, 2, 2, TO_DATE('20/10/2021', 'DD/MM/YYYY'),
120, '19:00', 'ACTIV');
INSERT INTO events_lco(event_id, play_id, venue_id, event_date, price,
hour)
VALUES(events_lco_seq.NEXTVAL, 2, 2, TO_DATE('20/12/2021', 'DD/MM/YYYY'),
120, '19:00');

SELECT * FROM events_lco;
```

Venues_lco

```
CREATE TABLE venues_lco
    (venue_id NUMBER(5) PRIMARY KEY,
     venue_name VARCHAR2(30) NOT NULL,
     seat_capacity NUMBER(4) NOT NULL CHECK(seat_capacity > 10),
     address_id NUMBER(5) REFERENCES address_venues_lco(address_id) ON
DELETE CASCADE);


CREATE SEQUENCE venues_lco_seq
START WITH 1
INCREMENT BY 1;

INSERT INTO venues_lco(venue_id, venue_name, seat_capacity, address_id)
VALUES(venues_lco_seq.NEXTVAL, 'Sala Pictura', 150, 2);
INSERT INTO venues_lco(venue_id, venue_name, seat_capacity, address_id)
VALUES(venues_lco_seq.NEXTVAL, 'Ion Caramitru', 940, 2);
INSERT INTO venues_lco(venue_id, venue_name, seat_capacity, address_id)
VALUES(venues_lco_seq.NEXTVAL, 'Sala Mica', 20, 2);
INSERT INTO venues_lco(venue_id, venue_name, seat_capacity, address_id)
VALUES(venues_lco_seq.NEXTVAL, 'Sala Mica', 40, 3);
INSERT INTO venues_lco(venue_id, venue_name, seat_capacity, address_id)
VALUES(venues_lco_seq.NEXTVAL, 'Sala Mica', 50, 4);
SELECT * FROM venues_lco;
```

Address_venues_lco

```
CREATE TABLE address_venues_lco
    (address_id NUMBER(5) PRIMARY KEY,
    institution_name VARCHAR2(30) NOT NULL,
    city VARCHAR2(20) NOT NULL);

CREATE SEQUENCE address_venues_lco_seq
START WITH 1
INCREMENT BY 1;

INSERT INTO address_venues_lco
VALUES(address_venues_lco_seq.NEXTVAL, 'Teatrul Nottara', 'Bucuresti');
INSERT INTO address_venues_lco
```

```
VALUES(address_venues_lco_seq.NEXTVAL, 'Teatrul National Bucuresti',
'Bucuresti');
INSERT INTO address_venues_lco
VALUES(address_venues_lco_seq.NEXTVAL, 'Teatrul Odeon', 'Bucuresti');
INSERT INTO address_venues_lco
VALUES(address_venues_lco_seq.NEXTVAL, 'Teatrul National', 'Iasi');
INSERT INTO address_venues_lco
VALUES(address_venues_lco_seq.NEXTVAL, 'Teatrul National Cluj-Napoca',
'Cluj-Napoca');
SELECT * FROM address_venues_lco;
```

Orders_lco

```
CREATE TABLE orders_lco
    (event_id NUMBER(5) REFERENCES events_lco(event_id) ON DELETE CASCADE,
     customer_id NUMBER(5) REFERENCES customers_lco(customer_id) ON DELETE
CASCADE,
     num_seats NUMBER(2) NOT NULL CHECK(num_seats > 0));

INSERT INTO orders_lco(event_id, customer_id, num_seats)
VALUES(1, 3, 3);
INSERT INTO orders_lco(event_id, customer_id, num_seats)
VALUES(1, 4, 5);
INSERT INTO orders_lco(event_id, customer_id, num_seats)
VALUES(3, 2, 4);
INSERT INTO orders_lco(event_id, customer_id, num_seats)
VALUES(4, 1, 1);
INSERT INTO orders_lco(event_id, customer_id, num_seats)
VALUES(4, 2, 2);
INSERT INTO orders_lco(event_id, customer_id, num_seats)
VALUES(6, 1, 4);
INSERT INTO orders_lco(event_id, customer_id, num_seats)
VALUES(5, 2, 2);
INSERT INTO orders_lco(event_id, customer_id, num_seats)
VALUES(6, 5, 9);
INSERT INTO orders_lco(event_id, customer_id, num_seats)
VALUES(5, 1, 1);
INSERT INTO orders_lco(event_id, customer_id, num_seats)
VALUES(5, 5, 2);

SELECT * FROM orders_lco;
```

Customers_lco

```
CREATE TABLE customers_lco
    (customer_id NUMBER(5) PRIMARY KEY,
     first_name VARCHAR2(20) NOT NULL,
     last_name VARCHAR2(20) NOT NULL,
     email VARCHAR2(20) NOT NULL UNIQUE,
     phone_number VARCHAR2(10) UNIQUE);

CREATE SEQUENCE customers_lco_seq
START WITH 1
INCREMENT BY 1;

INSERT INTO customers_lco(customer_id, first_name, last_name, email,
phone_number)
VALUES(customers_lco_seq.NEXTVAL, 'Ioana', 'Pop', 'ioanapop@gmail.com',
'0721174482');
INSERT INTO customers_lco(customer_id, first_name, last_name, email)
VALUES(customers_lco_seq.NEXTVAL, 'Victor', 'Avram',
'avram_vic@gmail.com');
INSERT INTO customers_lco(customer_id, first_name, last_name, email,
phone_number)
VALUES(customers_lco_seq.NEXTVAL, 'Marius', 'Dediu', 'marius@gmail.com',
'0787123832');
INSERT INTO customers_lco(customer_id, first_name, last_name, email,
phone_number)
VALUES(customers_lco_seq.NEXTVAL, 'Erika', 'Pitaru', 'pit_erika@gmail.com',
'0719921004');
INSERT INTO customers_lco(customer_id, first_name, last_name, email,
phone_number)
VALUES(customers_lco_seq.NEXTVAL, 'Alexandra', 'Popa',
'alex_pop@gmail.com', '0721193392');
SELECT * FROM customers_lco;
```

g. Scrierea interogărilor pentru baza de date

1. Platforma creează un sistem de puncte, oferind puncte bonus (care ulterior pot fi folosite la reduceri ale prețului biletelor) pentru următoarele categorii:

- 20 pct. pentru persoana cu cele mai multe bilete cumpărate

- 10 pct. dacă a cumparat cel puțin 5 bilete în total și cel puțin 2 sunt pe anul curent
- 2 pct. Dacă nu se încadrează în celelalte două categorii, dar au persoanele care au numărul de telefon în rețeaua Vodafone

Să se afișeze persoanele care îndeplinesc categoriile, împreună cu numărul de bilete cumpărate și numărul de punct bonus.
Rezolvarea conține: CASE, LEFT JOIN, SUM, INSTR, EXTRACT, subcerere în clauza FROM

```
SELECT first_name, last_name, phone_number, tickets,
CASE
WHEN (ROWNUM = 1) THEN 20
WHEN (tickets > 4 AND tickets_2021 >= 2) THEN 10
WHEN (phone_number IS NOT NULL AND INSTR(phone_number, '072') = 1)
        THEN 2
ELSE 0
END puncte_bonus
FROM (SELECT customer_id, first_name, last_name, phone_number,
SUM(num_seats) tickets
        FROM customers_lco
        JOIN orders_lco USING(customer_id)
        GROUP BY customer_id, first_name, last_name, phone_number
        ORDER BY tickets DESC) customers_ordered
LEFT JOIN (SELECT customer_id, SUM(num_seats) tickets_2021
        FROM orders_lco
        JOIN events_lco USING (event_id)
        WHERE EXTRACT(YEAR FROM event_date) = '2022'
        GROUP BY customer_id) USING (customer_id);
```

2. Datorită numărului crescut de cazuri Covid, se verifica posibilitatea mutării spectacolelor peste 3 luni, dacă îndeplinesc una din următoarele condiții:

- sunt din viitor și sunt anulate
- au mai multe bilete vândute decât jumătatea capacității sălii

Să se afișeze evenimentele care verifică condițiile, împreună cu data de peste 3 luni.
Rezolvarea conține: ADD_MONTHS, GROUP BY, HAVING, SUM

```
SELECT event_id, event_date, ADD_MONTHS(event_date, 3)
FROM events_lco ev
WHERE (status IS NOT NULL AND UPPER(status) = 'ANULAT' AND event_date >
sysdate) OR
event_id IN (SELECT event_id
             FROM orders_lco ord
```

```
            GROUP BY event_id
            HAVING SUM(num_seats) > (SELECT seat_capacity / 2
                                     FROM venues_lco
                                     JOIN events_lco USING (venue_id)
                                     WHERE event_id = ord.event_id));
```

3. Să se afișeze subalternii responsabilului grupei de teatru care a jucat în cele mai multe piese.
Daca sunt mai multi, se selecteaza cel mai bătrân.
Rezolvarea conține: START WITH, CONNECT BY, MAX, COUNT, GROUP BY, HAVING

```
SELECT LPAD(first_name || ' ' || last_name, LEVEL - 1 + LENGTH(first_name
            || ' ' || last_name)) nume
FROM actors_lco
START WITH actor_id = (SELECT actor_id
                       FROM (
                            SELECT actor_id
                            FROM actors_lco
                            WHERE representative_id IS NULL
                            AND actor_id IN (SELECT actor_id
                                        FROM plays_actors_lco
                                        GROUP BY actor_id
                                        HAVING COUNT(play_id) = (SELECT
                                                    MAX(COUNT(play_id))
                                                    FROM plays_actors_lco
                                                    GROUP BY actor_id))
                            ORDER BY date_of_birth)
                        WHERE ROWNUM = 1)
CONNECT BY PRIOR actor_id = representative_id;
```

4. Să se afișeze toate datele calendaristice dintre cele mai îndepărtate evenimente.
Rezolvarea conține: CROSS JOIN, CONNECT BY, TO_DATE, MIN, MAX

```
SELECT date_min + ROWNUM
FROM (SELECT MIN(event_date) date_min
        FROM events_lco)
CROSS JOIN (SELECT MAX(event_date) date_max
        FROM events_lco)
CONNECT BY ROWNUM < TO_DATE(date_max, 'DD/MM/YYYY') - TO_DATE(date_min,
'DD/MM/YYYY');
```

5. Să se afișeze clienții care au cumpărat cel puțin aceleași bilete ca cele cumpărate de clientul ‘Alexandra Popa’.
Rezolvarea conține: operatorul DIVISION, MINUS, INNER JOIN, INITCAP

```
SELECT *
FROM customers_lco cust
WHERE NOT EXISTS (SELECT event_id
                  FROM orders_lco
                  JOIN customers_lco
                  ON orders_lco.customer_id = customers_lco.customer_id
                  WHERE INITCAP(first_name) = 'Ioana'
                  AND INITCAP(last_name) = 'Pop'

                  MINUS

                  SELECT event_id
                  FROM orders_lco
                  WHERE customer_id = cust.customer_id)
AND INITCAP(first_name) != 'Alexandra' AND INITCAP(last_name) != 'Popa';
```

6. Să se afișeze actorii care au spectacole în 2022 neanulate și nu joacă singuri într-o piesă.
Rezolvarea conține: INTERSECT, INNER JOIN, UPPER, EXTRACT, subcerere în clauza WHERE

```
SELECT act.first_name, act.last_name, act.date_of_birth FROM actors_lco act
JOIN plays_actors_lco ON plays_actors_lco.actor_id = act.actor_id
JOIN plays_lco ON plays_actors_lco.play_id = plays_lco.play_id
JOIN events_lco ev ON plays_lco.play_id = ev.play_id
WHERE (ev.status IS NULL OR UPPER(ev.status) != 'ANULAT')
AND EXTRACT(YEAR FROM event_date) = '2022'

INTERSECT

SELECT act.first_name, act.last_name, act.date_of_birth FROM actors_lco act
JOIN plays_actors_lco pa ON pa.actor_id = act.actor_id
WHERE EXISTS (SELECT 1
FROM plays_actors_lco
WHERE actor_id <> act.actor_id AND pa.play_id = play_id);
```

Cozma Laura-Elena
BDTS, grupa 405

7. Să se afișeze clienții care au achiziționat la spectacole în minim două sali diferite în 2022 și spectacolele nu sunt anulate sau au cel puțin un bilet la un spectacol anulat în 2021.
Rezolvarea conține: UNION, INNER JOIN, GROUP BY, HAVING

```
SELECT customer_id, first_name, last_name
FROM customers_lco
JOIN orders_lco USING (customer_id)
JOIN events_lco USING (event_id)
WHERE (status IS NULL OR UPPER(status) != 'ANULAT') AND EXTRACT(YEAR FROM
event_date) = '2022'
GROUP BY customer_id, first_name, last_name
HAVING COUNT(DISTINCT venue_id) >= 2

UNION

SELECT customer_id, first_name, last_name FROM customers_lco
JOIN orders_lco USING (customer_id)
JOIN events_lco USING (event_id)
WHERE status IS NOT NULL AND UPPER(status) = 'ANULAT' AND EXTRACT(YEAR FROM
event_date) = '2021';
```

8.Să se afișeze pentru fiecare piesă de teatru, numărul de spectatori care vin în medie la acestea.
Rezolvarea conține: NVL, ROUND, AVG, SUM, GROUP BY, subcerere în FROM

```
SELECT title,
      (SELECT NVL(ROUND(AVG(total_seats), 1), 0)
       FROM (SELECT event_id, SUM(num_seats) total_seats
             FROM orders_lco
             GROUP BY event_id)
       JOIN events_lco USING(event_id)
       WHERE play_id = p.play_id) avg_seats,
      (SELECT NVL(AVG(price), 0)
       FROM events_lco
       WHERE play_id = p.play_id) avg_price
FROM plays_lco p;
```

9. Sa se afiseze piesele de teatru, pentru fiecare numărul de actori care joacă în piesă și numărul de evenimente. Piesele trebuie să aibă cel mai mare număr de evenimente în 2021.
Rezolvarea conține: TO_CHAR, MAX, COUNT, GROUP BY, HAVING subcerere în FROM

```
SELECT title, actors, COUNT(event_id) num_events
FROM plays_lco p
JOIN (SELECT play_id, COUNT(actor_id) actors
                FROM plays_actors_lco
                GROUP BY play_id) num_actors
      ON p.play_id = num_actors.play_id
JOIN events_lco ev ON ev.play_id = num_actors.play_id
WHERE TO_CHAR(event_date, 'YYYY') = '2021'
GROUP BY ev.play_id, title, actors
HAVING COUNT(event_id) = (SELECT MAX(COUNT(event_id))
                          FROM events_lco
                          WHERE TO_CHAR(event_date, 'YYYY') = '2021'
                          GROUP BY play_id);
```

10. Să se afișeze detalii despre evenimentele din cea mai populară sală (are cele mai multe bilete cumpărate în număr de clienți) și în a căror piesă de teatru joacă minim doi actori.
Rezolvarea conține: INNER JOIN, GROUP BY, HAVING, MAX, COUNT, subcerere în clauza WHERE.

```
SELECT * FROM events_lco
WHERE venue_id IN (SELECT venue_id
                   FROM events_lco ev
                   JOIN orders_lco ord ON ord.event_id = ev.event_id
                   GROUP BY venue_id
                   HAVING COUNT(ord.event_id) = (SELECT
                                        MAX(COUNT(ord.event_id))
                                        FROM orders_lco ord
                                        JOIN events_lco ev ON
                                        ord.event_id = ev.event_id
                                        GROUP BY venue_id))
AND play_id IN (SELECT play_id FROM plays_actors_lco
            GROUP BY play_id
            HAVING COUNT(actor_id) >= 2);
```

11. Să se afiseze salile și instituțiile din Bucuresti care au spectacole în mai puțin de o lună, împreună cu toți actorii care vor urca pe scenă în acea lună. Numele actorilor va fi afișat în formatul P. Nume.
Rezolvarea conține: SUBSTR, MONTHS_BETWEEN, LOWER, ORDER BY, JOIN

```
SELECT DISTINCT institution_name, venue_name,
       SUBSTR(actors_lco.first_name, 1, 1) || '. ' || actors_lco.last_name
       actor
FROM address_venues_lco
JOIN venues_lco USING (address_id)
JOIN events_lco USING (venue_id)
JOIN plays_lco USING (play_id)
JOIN plays_actors_lco USING (play_id)
JOIN actors_lco USING (actor_id)
WHERE MONTHS_BETWEEN(event_date, sysdate) <= 1
      AND (status IS NULL OR LOWER(status) != 'anulat')
      AND LOWER(city) = 'bucuresti'
ORDER BY institution_name, venue_name;
```

12. Să se afișeze piesele de teatru din Bucuresti împreună cu profitul acestora, în ordine descrescătoare. Se menționează că unele piese nu vor avea profit pentru ca nu au fost bilete vandute, iar alte piese vor fi anulate, dar ar putea totuși să aibă profit.
Rezolvarea conține: RIGHT JOIN, subcerere în FROM, DECODE, NVL, ORDER BY

```
SELECT event_id, title, NVL(price * sold_seats, 0) profit, DECODE(status,
NULL, 'ACTIV', status)
FROM (SELECT event_id, NVL(SUM(num_seats), 0) sold_seats
               FROM events_lco
               JOIN plays_lco USING (play_id)
               JOIN venues_lco USING (venue_id)
               JOIN address_venues_lco USING (address_id)
               LEFT JOIN orders_lco USING (event_id)
               WHERE UPPER(city) = 'BUCURESTI'
               GROUP BY event_id) ev_filtered
RIGHT JOIN events_lco USING(event_id)
JOIN plays_lco USING(play_id)
ORDER BY profit DESC;
```

13. Să se afișeze perechi de tipul (id eveniment: valoare, id sala: valoare) pentru toate evenimentele care vor avea loc între data curentă și ‘31/12/2021’, iar drept locație va fi Teatrul National Bucuresti.

Rezolvarea conține: DECODE, FULL JOIN, LEFT JOIN, TO_DATE, INITCAP

```
SELECT '(id eveniment: ' || DECODE(event_id, NULL, ' nu are evenimente',
event_id) || ', id sala: ' || DECODE(venue_id, NULL, ' nu are locatia
stabilita', venue_id) || ')' perechi
FROM events_lco
FULL JOIN venues_lco USING (venue_id)
LEFT JOIN address_venues_lco USING(address_id)
WHERE (event_id IS NULL
OR TO_DATE(event_date, 'DD/MM/YYYY') BETWEEN TO_DATE(SYSDATE, 'DD/MM/YYYY')
AND TO_DATE('31/12/2022', 'DD/MM/YYYY'))
AND (venue_id IS NULL
OR INITCAP(institution_name) = 'Teatrul National Bucuresti');
```

14. Să se afișeze informații despre toți clienții care au cumparat bilete pentru toate evenimentele (inclusiv cele anulate) din ianuarie 2022.
Rezolvarea conține: operatorul DIVISION, TO_CHAR, JOIN

```
SELECT DISTINCT cust_1.*
FROM orders_lco ord_1
JOIN customers_lco cust_1 ON ord_1.customer_id = cust_1.customer_id
WHERE NOT EXISTS (SELECT 1
                  FROM events_lco ev_2
                  WHERE TO_CHAR(event_date, 'MM/YYYY') = '01/2022'
                  AND NOT EXISTS (SELECT 1
                                  FROM orders_lco
                                  WHERE ev_2.event_id = event_id AND
ord_1.customer_id = customer_id));
```

15. Să se afișeze capacitatea rămasă a sălilor de teatru la evenimentele cele mai populare (cu cele mai multe bilete vândute).
Rezolvarea conține: GROUP BY, HAVING, JOIN, MAX, SUM

```
SELECT popular_ev.event_id, seat_capacity - scaune as "Capacitate ramasa"
FROM events_lco ev
JOIN venues_lco ven ON ev.venue_id = ven.venue_id
JOIN (SELECT event_id, SUM(num_seats) scaune
      FROM events_lco
      JOIN orders_lco USING (event_id)
      GROUP BY event_id
      HAVING SUM(num_seats) = (SELECT MAX(SUM(num_seats))
                               FROM orders_lco
```

```
                                GROUP BY event_id)) popular_ev
     ON ev.event_id = popular_ev.event_id;
```

16. Să se afișeze numărul de actori pentru piesele de teatru care sunt drame și au cel puțin un actor care nu e din București.
Rezolvarea conține: NULLIF, GROUP BY, LOWER, SUM, JOIN

```
SELECT p.play_id, p.title, COUNT(actor_id) num_act
FROM plays_lco p
JOIN plays_actors_lco pa ON (p.play_id = pa.play_id)
WHERE NULLIF(LOWER(genre), 'drama') IS NULL
AND EXISTS (SELECT
            actor_id
            FROM plays_actors_lco
            JOIN actors_lco USING (actor_id)
            WHERE play_id = p.play_id
            AND NULLIF(LOWER(place_of_birth), ‘bucuresti') IS NOT NULL)
GROUP BY p.play_id, p.title;
```

h. Crearea unui tabel de mesaje

```
CREATE TABLE messages_lco
(message_id NUMBER PRIMARY KEY,
message VARCHAR2(255),
message_type VARCHAR2(1) CHECK (message_type in ('E','W','I')),
created_by VARCHAR2(40) NOT NULL,
created_at DATE NOT NULL);

CREATE SEQUENCE messages_seq_lco
START WITH 1
INCREMENT BY 1;
```

Cozma Laura-Elena
BDTS, grupa 405

i. Ilustrarea noțiunilor de PL/SQL

Prima secvență de cod conține definirea pachetului. Pentru suprograme cerintele sunt următoarele:

1. Să se creeze o procedura care sa adauge in tabelul actors_lco o coloana plays, pentru fiecare actor afișându-se piesele în care joacă. Suprogramul sa intoarca numarul maxim de piese in care joaca un actor, impreuna cu actorii care se încadrează în această categorie.

2. Pentru piesa data ca parametru, sa se intoarca numărul de evenimente care au mai mult de 2 clienti și id-urile acestor evenimente.

3. Numele clientului care a platit cea mai mare suma pentru un spectacol dintr-o anumita sală dată ca parametru. Se vor trata cazurile de eroare cand nu se gasesc clienti sau se găsesc prea multi.

Pachetul conține și o procedură care se va ocupa de scrierea mesajelor în tabela messages.

```
CREATE OR REPLACE TYPE t_imbri_title_lco AS TABLE OF VARCHAR2(50);
/

ALTER TABLE actors_lco
ADD (plays t_imbri_title_lco)
NESTED TABLE plays STORE AS info_plays;

SELECT * FROM actors_lco;
ALTER TABLE actors_lco
DROP COLUMN plays;

CREATE OR REPLACE PACKAGE exercitii_proiect IS
    TYPE rec_act IS RECORD
        (cod_actor actors_lco.actor_id%TYPE,
         prenume actors_lco.first_name%TYPE,
         nume actors_lco.last_name%TYPE);
    TYPE t_index_rec IS TABLE OF rec_act INDEX BY BINARY_INTEGER;
    PROCEDURE actors_with_max_num_plays (t_rec OUT t_index_rec,
                                         num_max_plays OUT NUMBER);

    TYPE tab_events IS TABLE OF events_lco.event_id%TYPE;
```

```
    PROCEDURE num_events_for_play(v_play_id IN NUMBER,
                                  num_events OUT NUMBER,
                                  t OUT tab_events);

    FUNCTION max_price_for_location(id_v venues_lco.venue_id%TYPE)
    RETURN VARCHAR2;

    PROCEDURE set_error_messages_lco(error_message IN VARCHAR2, msg_type IN
VARCHAR2);
END exercitii_proiect;
/


CREATE OR REPLACE PACKAGE BODY exercitii_proiect IS

    PROCEDURE actors_with_max_num_plays (t_rec OUT t_index_rec,
num_max_plays OUT NUMBER)
    IS
        TYPE tab_index_actors IS TABLE OF rec_act INDEX BY BINARY_INTEGER;
        t_act tab_index_actors;
        t_title t_imbri_title_lco;
        idx NUMBER := 0;
        i NUMBER;
    BEGIN
        num_max_plays := 0;
        SELECT MAX(COUNT(play_id)) INTO num_max_plays
        FROM plays_actors_lco
        GROUP BY actor_id;

        SELECT actor_id, first_name, last_name BULK COLLECT INTO t_act
        FROM actors_lco;

        i := t_act.FIRST;
        WHILE (i <= t_act.LAST) LOOP
            SELECT title BULK COLLECT INTO t_title
            FROM plays_lco
            JOIN plays_actors_lco USING (play_id)
            WHERE actor_id = t_act(i).cod_actor;

            UPDATE actors_lco
            SET plays = t_title
            WHERE actor_id = t_act(i).cod_actor;
```

```
            IF t_title.COUNT = num_max_plays THEN
                idx := idx + 1;
                t_rec(idx) := t_act(i);
            END IF;
            i:= t_act.NEXT(i);
        END LOOP;
    END actors_with_max_num_plays;


    PROCEDURE num_events_for_play(v_play_id IN NUMBER, num_events OUT
NUMBER, t OUT tab_events)
    IS
        ck NUMBER := -1;
        exceptie EXCEPTION;
        error_msg VARCHAR2(100);
        CURSOR c_ev IS
            (SELECT ord.event_id
            FROM events_lco ev
            JOIN orders_lco ord ON (ord.event_id = ev.event_id)
            WHERE ev.play_id = v_play_id
            GROUP BY ord.event_id
            HAVING COUNT(DISTINCT ord.customer_id) >= 2);
    BEGIN
        SELECT COUNT(*) INTO ck
        FROM plays_lco
        WHERE play_id = v_play_id;

        IF ck = 0 THEN
            RAISE exceptie;
        END IF;

        IF c_ev%ISOPEN THEN
            CLOSE c_ev;
            exercitii_proiect.set_error_messages_lco('Ati uitat cursorul
deschis', 'W');
        END IF;

        OPEN c_ev;
        FETCH c_ev BULK COLLECT INTO t;
        num_events := t.COUNT ;
        CLOSE c_ev;

    EXCEPTION
```

```
        WHEN exceptie THEN
            exercitii_proiect.set_error_messages_lco('Nu exista piesa cu
id-ul specificat', 'E');
            RAISE_APPLICATION_ERROR(-20210, 'Nu exista piesa cu id-ul
specificat');
        WHEN OTHERS THEN
            error_msg := SUBSTR(SQLERRM,1,100);
            exercitii_proiect.set_error_messages_lco(error_msg, 'E');
            RAISE_APPLICATION_ERROR(-20001, error_msg);
    END num_events_for_play;


    FUNCTION max_price_for_location(id_v venues_lco.venue_id%TYPE)
    RETURN VARCHAR2
    IS
        ck NUMBER := -1;
        exceptie EXCEPTION;
        error_msg VARCHAR2(200);
        nume VARCHAR2(100);
        max_sum NUMBER(5) := 0;
    BEGIN

        SELECT COUNT(*) INTO ck
        FROM venues_lco
        WHERE venue_id = id_v;

        IF ck = 0 THEN
            RAISE exceptie;
        END IF;

        SELECT MAX(MAX(num_seats * price)) INTO max_sum
        FROM orders_lco ord
        JOIN events_lco ev ON (ord.event_id = ev.event_id)
        WHERE venue_id = id_v
        GROUP BY customer_id;
        DBMS_OUTPUT.PUT_LINE('Suma maxima: ' || max_sum);
        SELECT last_name INTO nume
        FROM customers_lco cust
        JOIN orders_lco ord ON (cust.customer_id = ord.customer_id)
        JOIN events_lco ev ON (ord.event_id = ev.event_id)
        WHERE venue_id = id_v
        GROUP BY ord.customer_id, last_name
        HAVING MAX(num_seats * price) = max_sum;
```

```
        RETURN nume;

        EXCEPTION
            WHEN exceptie THEN
                exercitii_proiect.set_error_messages_lco('Nu exista sala cu
id-ul specificat', 'E');
                RAISE_APPLICATION_ERROR(-20220, 'Nu exista sala cu id-ul
specificat');
            WHEN NO_DATA_FOUND THEN
                error_msg := SUBSTR(SQLERRM,1,100);
                exercitii_proiect.set_error_messages_lco(error_msg, 'E');
                RAISE_APPLICATION_ERROR(-20001, error_msg);
            WHEN TOO_MANY_ROWS THEN
                error_msg := SUBSTR(SQLERRM,1,100);
                exercitii_proiect.set_error_messages_lco(error_msg, 'E');
                RAISE_APPLICATION_ERROR(-20001, error_msg);
            WHEN OTHERS THEN
                error_msg := SUBSTR(SQLERRM,1,100);
                exercitii_proiect.set_error_messages_lco(error_msg, 'E');
                RAISE_APPLICATION_ERROR(-20001, error_msg);
    END max_price_for_location;

    PROCEDURE set_error_messages_lco(error_message IN VARCHAR2, msg_type IN
VARCHAR2)
    IS
        PRAGMA AUTONOMOUS_TRANSACTION;
    BEGIN
        INSERT INTO messages_lco
        VALUES(messages_seq_lco.NEXTVAL, error_message, msg_type,
SYS.LOGIN_USER, SYSDATE);
        COMMIT;
    END;

END exercitii_proiect;
/
```

Apelare cerința 1:

```
SET SERVEROUTPUT ON;

DECLARE
t_rec exercitii_proiect.t_index_rec;
num_max_plays NUMBER;

BEGIN
    exercitii_proiect.actors_with_max_num_plays(t_rec, num_max_plays);
    DBMS_OUTPUT.PUT_LINE('Numarul maxim de piese este: ' || num_max_plays);
    DBMS_OUTPUT.PUT_LINE('Actorii cu numar maxim de piese sunt:');
    FOR i IN t_rec.FIRST..t_rec.LAST LOOP
        IF t_rec.exists(i) THEN
            DBMS_OUTPUT.PUT_LINE(t_rec(i).cod_actor || ' ' ||
t_rec(i).prenume || ' ' || t_rec(i).nume);
        END IF;
    END LOOP;
END;
/

SELECT * FROM actors_lco;
```

Apelare cerința 2:

```
SELECT * FROM messages_lco;
DECLARE
    nr_ev NUMBER(5);
    t_ev exercitii_proiect.tab_events;
BEGIN
    -- Apel exceptie nu a fost gasita piesa cu id-ul respectiv
    exercitii_proiect.num_events_for_play(14, nr_ev, t_ev);
    -- Apel piesa cu evenimente
    -- exercitii_proiect.num_events_for_play(2, nr_ev, t_ev);
    DBMS_OUTPUT.PUT_LINE('Numarul de evenimente pentru piesa cu id-ul dat
ca parametru cu minim 2 clienti: ' || nr_ev);
    IF (t_ev.COUNT > 0) THEN
        DBMS_OUTPUT.PUT_LINE('Id-urile evenimentelor sunt:');
        FOR i IN t_ev.FIRST..t_ev.LAST LOOP
            IF t_ev.EXISTS(i) THEN
                DBMS_OUTPUT.PUT_LINE(t_ev(i));
            END IF;
```

```
        END LOOP;
    END IF;

END;
/
```

Apelare cerința 3:

```
SELECT * FROM orders_lco;
BEGIN
    -- Nu exista venue
    DBMS_OUTPUT.PUT_LINE(exercitii_proiect.max_price_for_location(23));
    -- No data found:
    -- DBMS_OUTPUT.PUT_LINE(exercitii_proiect.max_price_for_location(3));
    --DBMS_OUTPUT.PUT_LINE(exercitii_proiect.max_price_for_location(1));
END;
/
```

Triggeri:

Trigger de tip LMD la nivel de comanda si trigger de tip LMD la nivel de linie
Să se creeze un trigger care se declanşează dacă se rezervă mai multe locuri într-o sală decât capacitatea sălii (cazul mutating table).

```
CREATE OR REPLACE PACKAGE pkt_max_seats_lco
AS
TYPE t_idx_cod_event IS TABLE OF events_lco.event_id%TYPE INDEX BY
BINARY_INTEGER;
TYPE t_idx_cod_client IS TABLE OF customers_lco.customer_id%TYPE INDEX BY
BINARY_INTEGER;
t_cod_event t_idx_cod_event;
t_cod_client t_idx_cod_client;
v_nr_elem BINARY_INTEGER := 0;
END pkt_max_seats_lco;
/
DROP PACKAGE pkt_max_seats;
```

```
-- trigger la nivel de linie
CREATE OR REPLACE TRIGGER trig_max_seats_linie
BEFORE INSERT OR UPDATE OF num_seats, event_id ON orders_lco
FOR EACH ROW
BEGIN
pkt_max_seats_lco.v_nr_elem := pkt_max_seats_lco.v_nr_elem + 1;
pkt_max_seats_lco.t_cod_event(pkt_max_seats_lco.v_nr_elem) :=
:NEW.event_id;
END;
/


CREATE OR REPLACE TRIGGER trig_max_seats_instr
AFTER INSERT OR UPDATE OF num_seats, event_id ON orders_lco
DECLARE
v_cod_event events_lco.event_id%TYPE;
v_nr_seats NUMBER;
v_max_seats NUMBER;
BEGIN
FOR i IN 1..pkt_max_seats_lco.v_nr_elem LOOP
    v_cod_event := pkt_max_seats_lco.t_cod_event(i);
    SELECT SUM(num_seats) INTO v_nr_seats
    FROM orders_lco
    WHERE event_id = v_cod_event;

    SELECT seat_capacity INTO v_max_seats
    FROM venues_lco
    JOIN events_lco USING(venue_id)
    WHERE event_id = v_cod_event;

    IF v_nr_seats > v_max_seats THEN
        exercitii_proiect.set_error_messages_lco('Numarul de scaune
rezervate depaseste capacitatea ramasa a salii', 'E');
        RAISE_APPLICATION_ERROR(-20230, 'Numarul de scaune rezervate
depaseste capacitatea ramasa a salii');
    END IF;
END LOOP;

pkt_max_seats_lco.v_nr_elem := 0;
END;
/
```

```
ROLLBACK;
SELECT * FROM orders_lco;
ALTER TRIGGER trig_max_seats_instr DISABLE;

-- Nu da eroare
UPDATE orders_lco
SET num_seats = 9
WHERE event_id = 1
AND customer_id = 3;

-- Da eroare
UPDATE orders_lco
SET num_seats = 99
WHERE event_id = 1
AND customer_id = 3;

-- Eroare
UPDATE orders_lco
SET num_seats = 51
WHERE event_id = 1
AND customer_id = 3;
-- Eroare aici
UPDATE orders_lco
SET event_id = 1
WHERE event_id = 4
AND customer_id = 1;
```

Trigger de tip LMD la nivel de linie
Nu se permite modificarea pretului biletelor daca s-au cumparat deja bilete la eveniment.

```
CREATE OR REPLACE TRIGGER check_for_price_updates
BEFORE UPDATE OF price ON events_lco
FOR EACH ROW
DECLARE
nr NUMBER := 0;
BEGIN
SELECT COUNT(customer_id) INTO nr
FROM orders_lco
WHERE event_id = :NEW.event_id;
```

```
IF (nr != 0) AND (:OLD.price != :NEW.price) THEN
    exercitii_proiect.set_error_messages_lco('Valoarea unui bilet nu poate
fi modificata', 'E');
    RAISE_APPLICATION_ERROR (-20240, 'Valoarea unui bilet nu poate fi
modificata');

END IF;
END;
/

SELECT * FROM orders_lco;

-- Se poate modifica
UPDATE events_lco
SET price = 100
WHERE event_id = 2;

-- Nu se poate modifica
UPDATE events_lco
SET price = 120
WHERE event_id = 4;
ROLLBACK;
SELECT * FROM messages_lco ORDER BY message_id;
```

Trigger de tip LDD care se declanșează la CREATE, DROP sau ALTER pe schemă și inserează informația în tabelul messages.

```
CREATE OR REPLACE TRIGGER trig_alter_drop_table
AFTER CREATE OR ALTER OR DROP  ON SCHEMA
BEGIN
    DBMS_OUTPUT.PUT_LINE('The LDD trigger was called!');
    exercitii_proiect.set_error_messages_lco('Create, alter or drop was
performed on schema', 'I');
END;
/
```