Tema 7 - Laborator

Aelenei Alex

Cuprins

1 Introducere **2**

2 Exercițiul 4 **2**

1 Introducere

Diagrama conceptuală și diagrama entitate-relație utilizate în această temă sunt mai jos. Datele de test folosite sunt cele utilizate în Tema 1 și Tema 2.

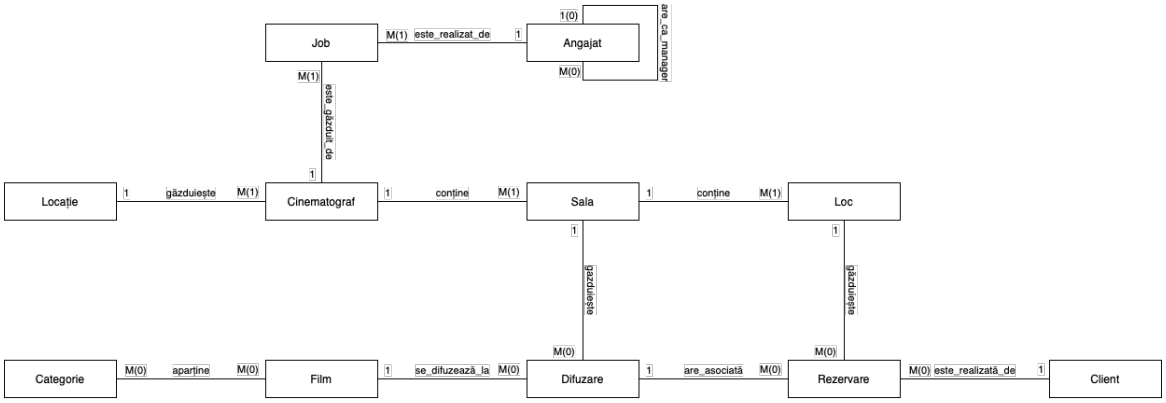

Figura 1: Diagrama conceptuală

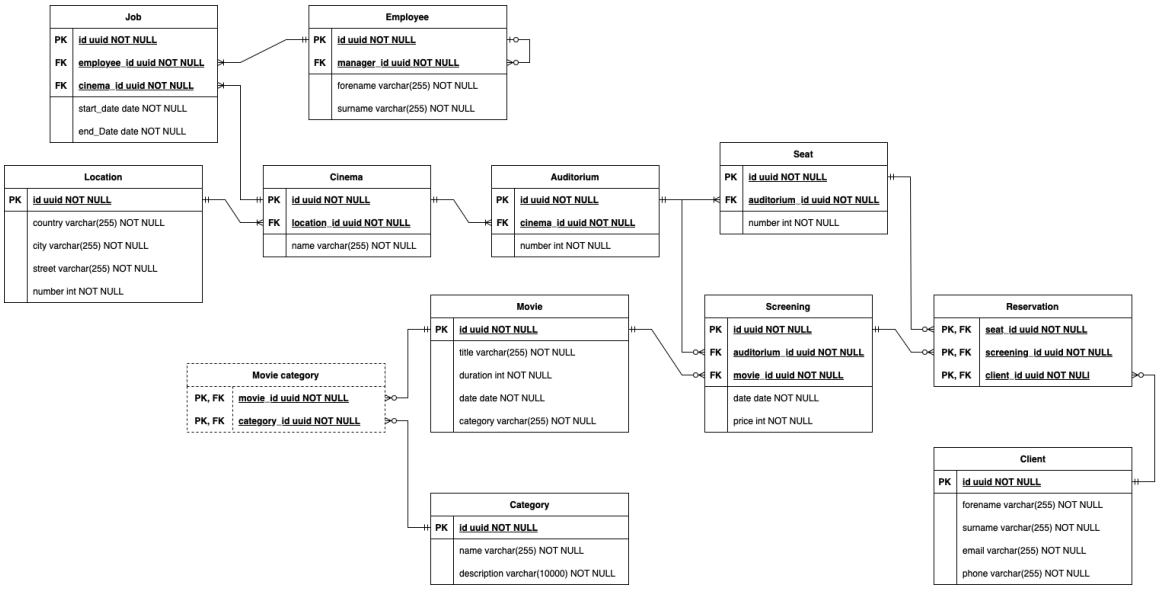

Figura 2: Diagrama entitate-relație

Aceste query-uri au fost rulate pe o instanță de Oracle Database 19, bazată pe imaginea oficială de la Oracle, dar cu un build pentru MacOS ARM local.

2 Exercițiul 4

Problema 3 din *Laborator5_PLSQL.pdf* ne cere să rezolvăm următoarea cerință:

- Creați tabelul info_dept_*** cu următoarele coloane:
  - id (codul departamentului) – cheie primară;
  - nume_dept (numele departamentului);
  - plati (suma alocată pentru plata salariilor angajaților care lucrează în departamentul respectiv).
- Introduceți date în tabelul creat anterior corespunzătoare informațiilor existente în schemă.
- Definiți un declanșator care va actualiza automat câmpul plati atunci când se introduce un nou salariat, respectiv se șterge un salariat sau se modifică salariul unui angajat.

Cerința adaptată la schema bazei de date utilizate va fi următoarea:

- Creați tabelul cinema_revenue cu următoarele coloane:
  - cinema_id (codul cinemaului) – cheie primară;
  - name (numele cinemaului);
  - revenue (venitul generat de cinemaul respectiv din rezervările făcute la difuzările filmelor).
- Introduceți date în tabelul creat anterior corespunzătoare informațiilor existente în schemă.
- Definiți un declanșator care va actualiza automat câmpul revenue atunci când se introduce o nouă rezervare, când se șterge o rezervare existentă, sau se schimbă o rezervare.

În rezolvarea acestei cerințe, am optat să adaug trei funcții ajutătoare, care să obțină datele relevante actualizării care trebuie făcută și să realizeze actualizarea.

```
01 |  CREATE TABLE CINEMA_REVENUE (
02 |      CINEMA_ID NUMBER(10) NOT NULL,
03 |      NAME VARCHAR2(255) NOT NULL,
04 |      REVENUE NUMBER(10) NOT NULL,
05 |      CONSTRAINT CINEMA_REVENUE_PK PRIMARY KEY (CINEMA_ID)
06 |  );
```

Listing 1: Query pentru definirea tabelului

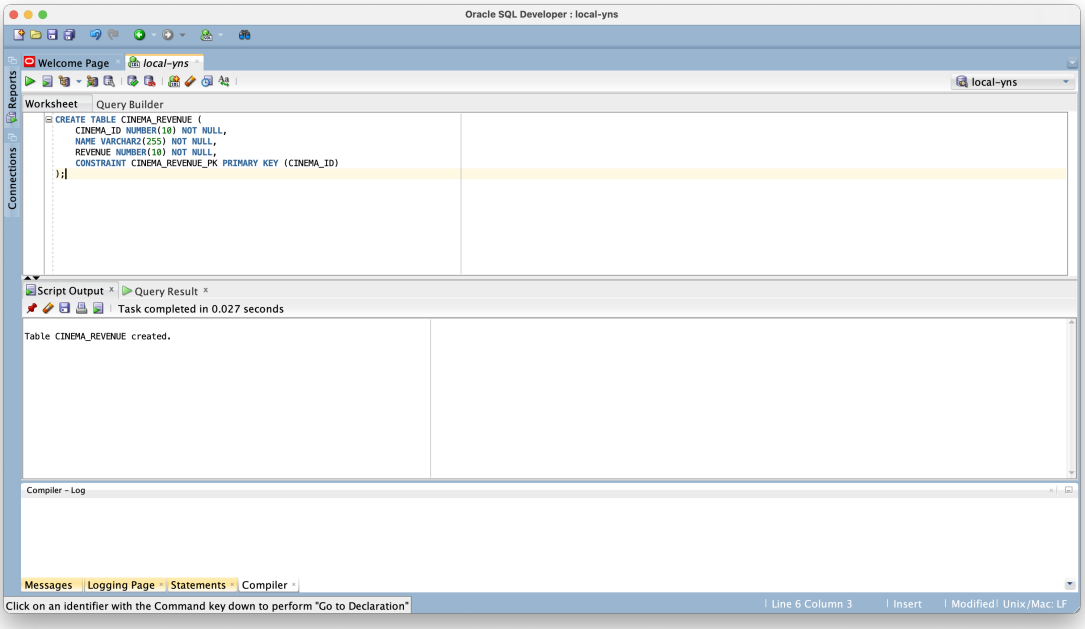

Figura 3: Rezultatul query-ului pentru definirea tabelului

```sql
INSERT INTO CINEMA_REVENUE (
    SELECT
        C.CINEMA_ID,
        C.NAME,
        SUM(SP.REVENUE) AS REVENUE
    FROM
        (
            SELECT
                S.SCREENING_ID,
                S.AUDITORIUM_ID,
                COUNT(R.SEAT_ID) * S.PRICE AS REVENUE
            FROM
                SCREENING   S
                INNER JOIN RESERVATION R
                ON R.SCREENING_ID = S.SCREENING_ID
            GROUP BY
                S.SCREENING_ID,
                S.PRICE,
                S.AUDITORIUM_ID
        )           SP
        INNER JOIN AUDITORIUM A
        ON A.AUDITORIUM_ID = SP.AUDITORIUM_ID
        INNER JOIN CINEMA C
        ON C.CINEMA_ID = A.CINEMA_ID
    GROUP BY
        C.CINEMA_ID,
        C.NAME
);
```

Listing 2: Query pentru inițializare

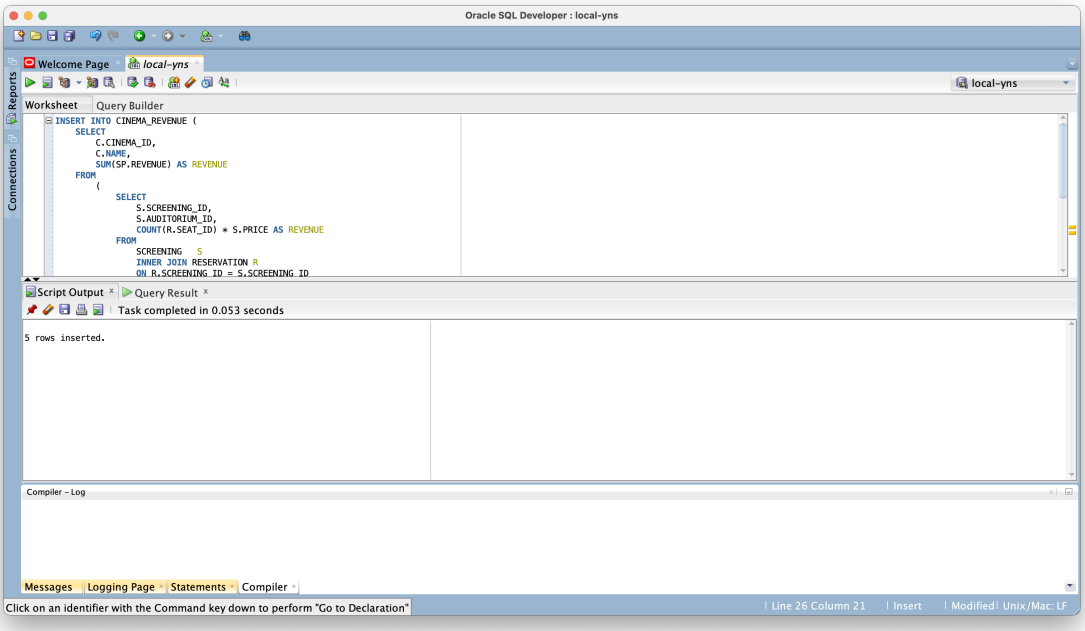

Figura 4: Rezultatul query-ului pentru inițializarea tabelului

```sql
CREATE TABLE CINEMA_REVENUE AS (
    SELECT
        C.CINEMA_ID,
        C.NAME,
        SUM(SP.REVENUE) AS REVENUE
    FROM
        (
            SELECT
                S.SCREENING_ID,
                S.AUDITORIUM_ID,
                COUNT(R.SEAT_ID) * S.PRICE AS REVENUE
            FROM
                SCREENING   S
                INNER JOIN RESERVATION R
                ON R.SCREENING_ID = S.SCREENING_ID
            GROUP BY
                S.SCREENING_ID,
                S.PRICE,
                S.AUDITORIUM_ID
        )           SP
        INNER JOIN AUDITORIUM A
        ON A.AUDITORIUM_ID = SP.AUDITORIUM_ID
        INNER JOIN CINEMA C
        ON C.CINEMA_ID = A.CINEMA_ID
    GROUP BY
        C.CINEMA_ID,
        C.NAME
);
```

Listing 3: Query pentru definirea tabelului și inițializare

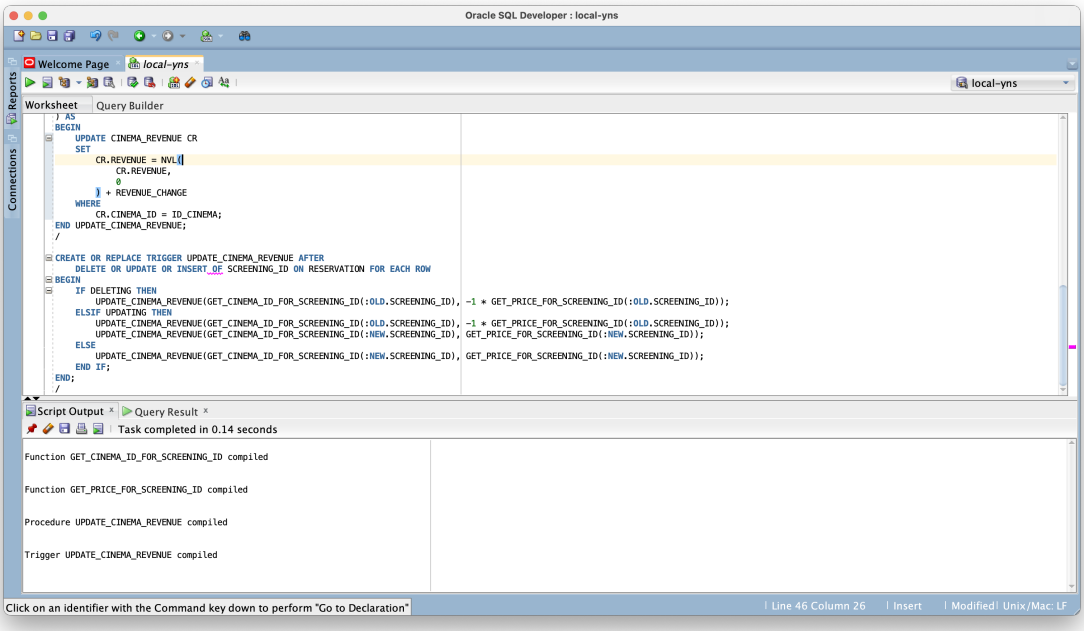

Figura 5: Rezultatul query-ului pentru definirea și inițializarea tabelului

```
CREATE OR REPLACE FUNCTION GET_CINEMA_ID_FOR_SCREENING_ID (
    ID_SCREENING SCREENING.SCREENING_ID%TYPE
) RETURN CINEMA.CINEMA_ID%TYPE IS
    ID_AUDITORIUM AUDITORIUM.AUDITORIUM_ID%TYPE;
    ID_CINEMA     CINEMA.CINEMA_ID%TYPE;
    CNT           NUMBER;
BEGIN
    SELECT
        S.AUDITORIUM_ID INTO ID_AUDITORIUM
    FROM
        SCREENING S
    WHERE
        S.SCREENING_ID = ID_SCREENING;
    SELECT
        A.CINEMA_ID INTO ID_CINEMA
    FROM
        AUDITORIUM A
    WHERE
        A.AUDITORIUM_ID = ID_AUDITORIUM;
    RETURN ID_CINEMA;
END GET_CINEMA_ID_FOR_SCREENING_ID;
/

CREATE OR REPLACE FUNCTION GET_PRICE_FOR_SCREENING_ID (
    ID_SCREENING SCREENING.SCREENING_ID%TYPE
) RETURN SCREENING.PRICE%TYPE IS
    PRICE SCREENING.PRICE%TYPE;
BEGIN
    SELECT
        S.PRICE INTO PRICE
    FROM
        SCREENING S
    WHERE
        S.SCREENING_ID = ID_SCREENING;
    return PRICE;
END GET_PRICE_FOR_SCREENING_ID;
/

CREATE OR REPLACE PROCEDURE UPDATE_CINEMA_REVENUE(
    ID_CINEMA CINEMA.CINEMA_ID%TYPE,
    REVENUE_CHANGE CINEMA_REVENUE.REVENUE%TYPE
) AS
BEGIN
    UPDATE CINEMA_REVENUE CR
    SET
        CR.REVENUE = NVL(
            CR.REVENUE,
            0
        ) + REVENUE_CHANGE
    WHERE
        CR.CINEMA_ID = ID_CINEMA;
END UPDATE_CINEMA_REVENUE;
/

CREATE OR REPLACE TRIGGER UPDATE_CINEMA_REVENUE AFTER
    DELETE OR UPDATE OR INSERT OF SCREENING_ID ON RESERVATION FOR EACH ROW
BEGIN
    IF DELETING THEN
        UPDATE_CINEMA_REVENUE(GET_CINEMA_ID_FOR_SCREENING_ID(:OLD.SCREENING_ID), -1 *
    GET_PRICE_FOR_SCREENING_ID(:OLD.SCREENING_ID));
    ELSIF UPDATING THEN
        UPDATE_CINEMA_REVENUE(GET_CINEMA_ID_FOR_SCREENING_ID(:OLD.SCREENING_ID), -1 *
    GET_PRICE_FOR_SCREENING_ID(:OLD.SCREENING_ID));
        UPDATE_CINEMA_REVENUE(GET_CINEMA_ID_FOR_SCREENING_ID(:NEW.SCREENING_ID),
    GET_PRICE_FOR_SCREENING_ID(:NEW.SCREENING_ID));
    ELSE
        UPDATE_CINEMA_REVENUE(GET_CINEMA_ID_FOR_SCREENING_ID(:NEW.SCREENING_ID),
    GET_PRICE_FOR_SCREENING_ID(:NEW.SCREENING_ID));
    END IF;
END;
/
```

Listing 4: Query pentru definirea trigger-ului și a funcțiilor auxiliare

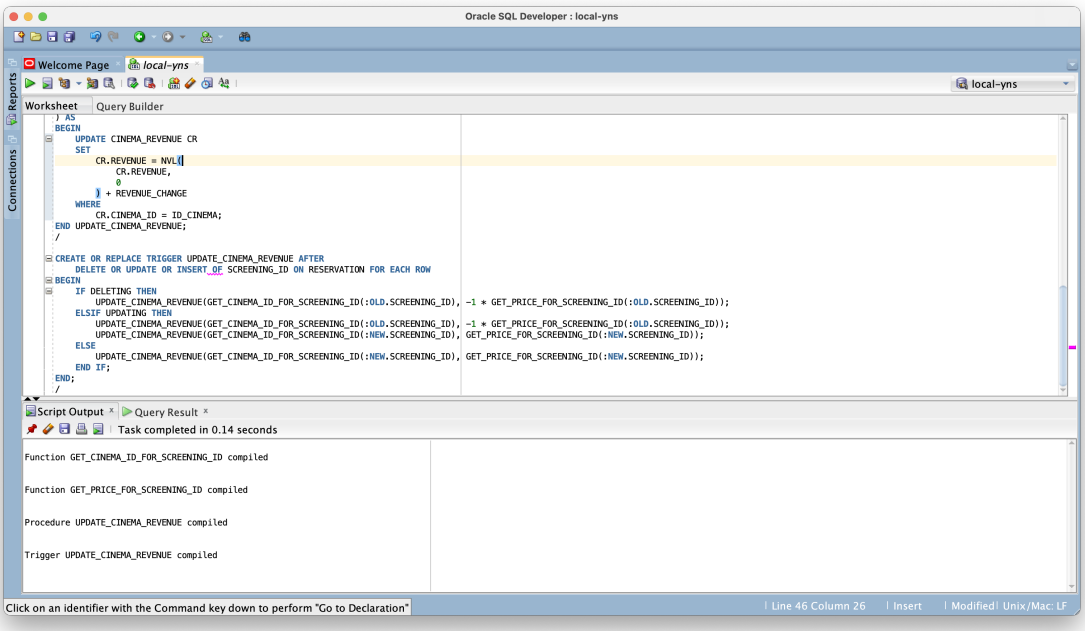

Figura 6: Rezultatul query-ului pentru trigger-ului și a funcțiilor auxiliare

```
01 |  SELECT
02 |      *
03 |  FROM
04 |      CINEMA_REVENUE;
05 |
06 |  INSERT INTO RESERVATION VALUES (
07 |      1,
08 |      1,
09 |      1
10 |  );
11 |
12 |  SELECT
13 |      *
14 |  FROM
15 |      CINEMA_REVENUE;
16 |
17 |  DELETE FROM RESERVATION R
18 |  WHERE
19 |      R.CLIENT_ID = 1
20 |      AND R.SCREENING_ID = 1
21 |      AND R.SEAT_ID = 1;
22 |
23 |  SELECT
24 |      *
25 |  FROM
26 |      CINEMA_REVENUE;
```

Listing 5: Query pentru testare

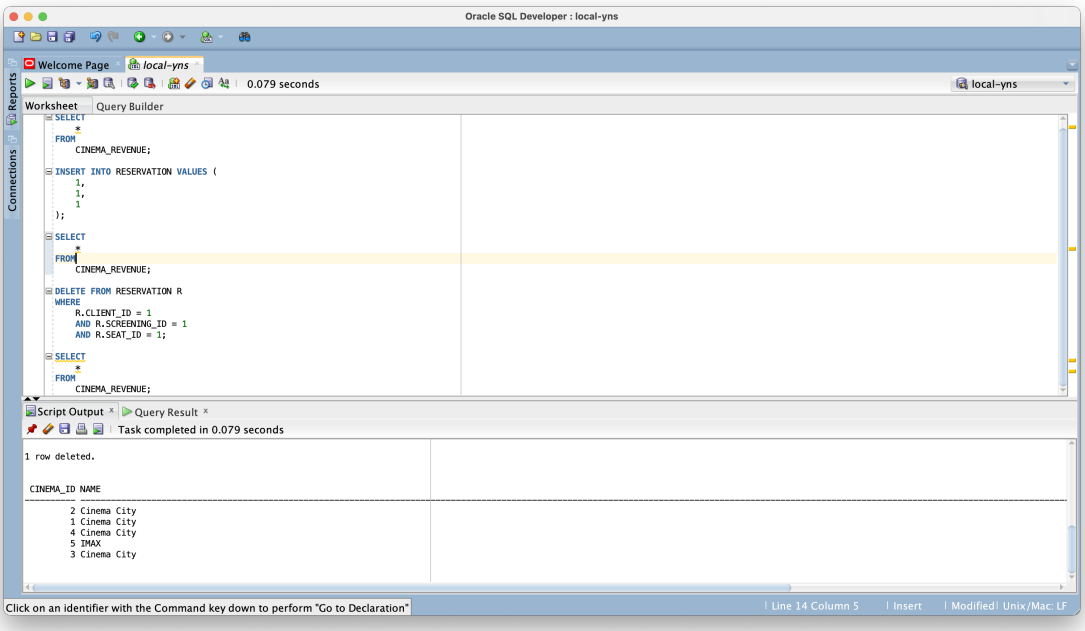

Figura 7: Rezultatul query-ului de testare

```
CINEMA_ID NAME                 REVENUE
---------- -------------------- ----------
         2 Cinema City                2339
         1 Cinema City                4568
         4 Cinema City                6700
         5 IMAX                       3737
         3 Cinema City                4497

1 row inserted.

CINEMA_ID NAME                 REVENUE
---------- -------------------- ----------
         2 Cinema City                2355
         1 Cinema City                4568
         4 Cinema City                6700
         5 IMAX                       3737
         3 Cinema City                4497

1 row deleted.

CINEMA_ID  NAME                 REVENUE
---------- -------------------- ----------
         2 Cinema City                2339
         1 Cinema City                4568
         4 Cinema City                6700
         5 IMAX                       3737
         3 Cinema City                4497
```

Listing 6: Rezultatul query-ului de testare

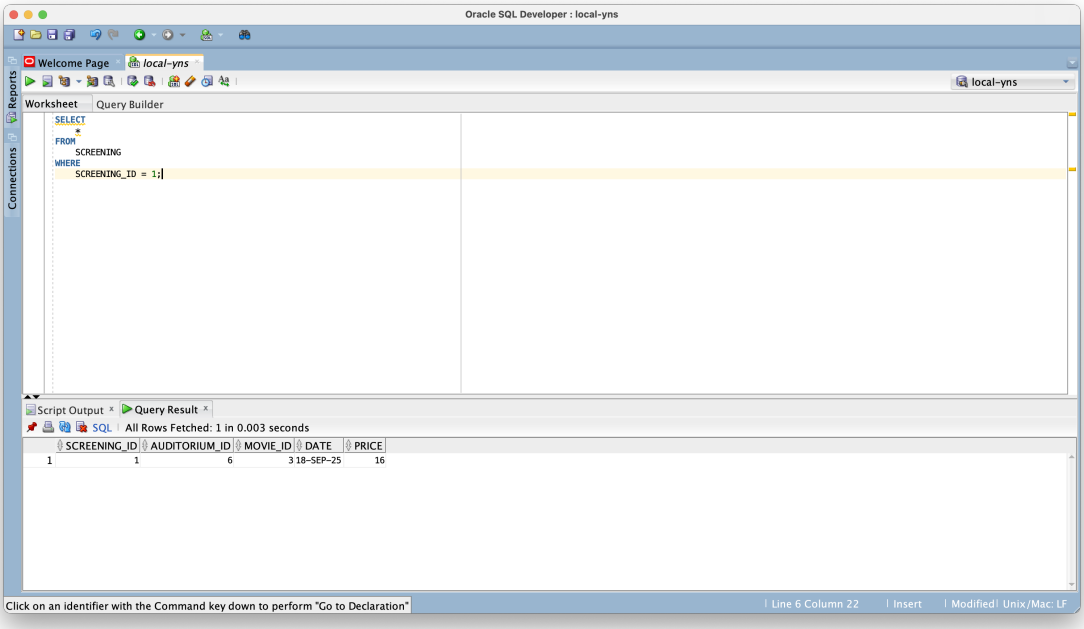

Figura 8: Conform datelor utilizate, rezervarea la difuzarea cu ID-ul 1 costa 16.