Dudau Claudia Maria – grupa 243

PROIECT SISTEME DE GESTIUNE A BAZELOR DE DATE

# Prezentare pe scurt a bazei de date

O baza de date ce conține informații despre seriale, producători, sezoane, episoade, actori si personaje, în scopul manipulării acestora.

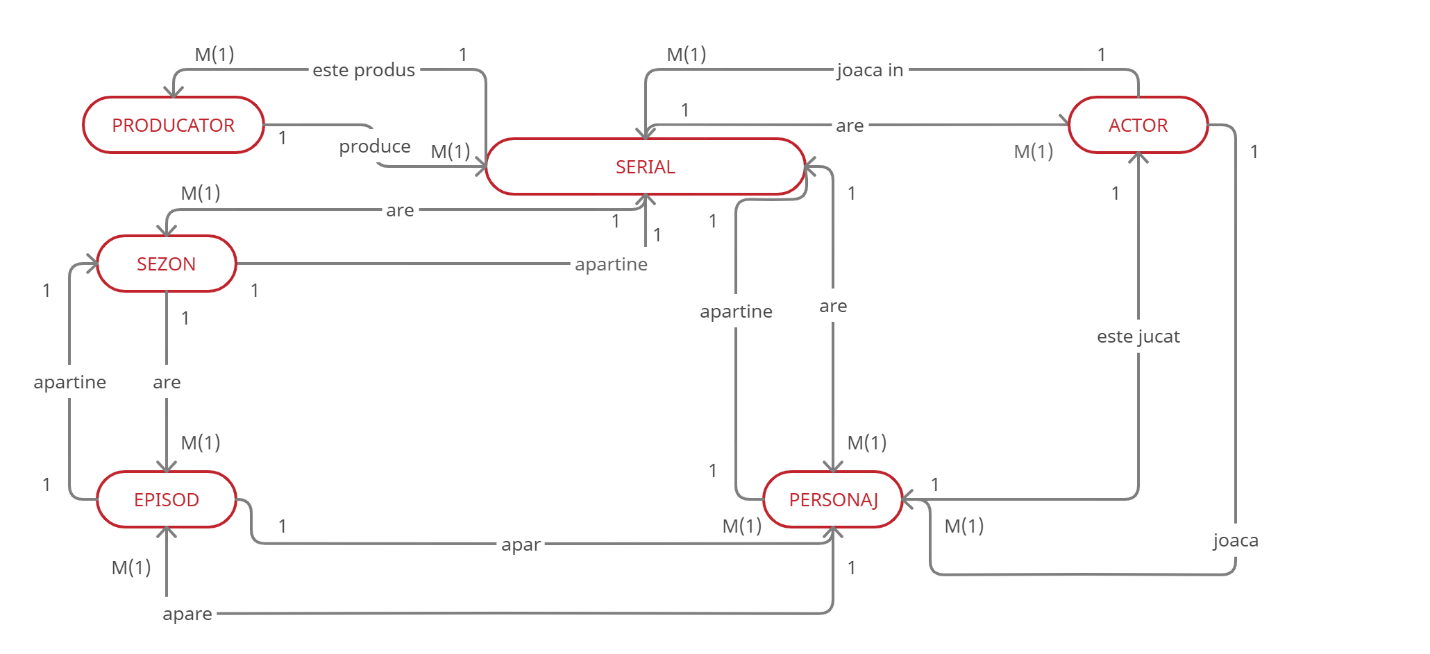
Ca orice baza de date, scopul său principal este acela de a oferi o structura logica a componentelor și a relațiilor dintre acestea pentru ca informațiile sa poată fi accesate cu ușurință.

Aceasta este utila in special site-urilor ce se ocupa cu oferirea de informații legate de diverse seriale (cum ar fi imdb), dar si pentru aplicații care se ocupa cu gestionarea datelor de acest fel (adăugare, modificare, ștergere).

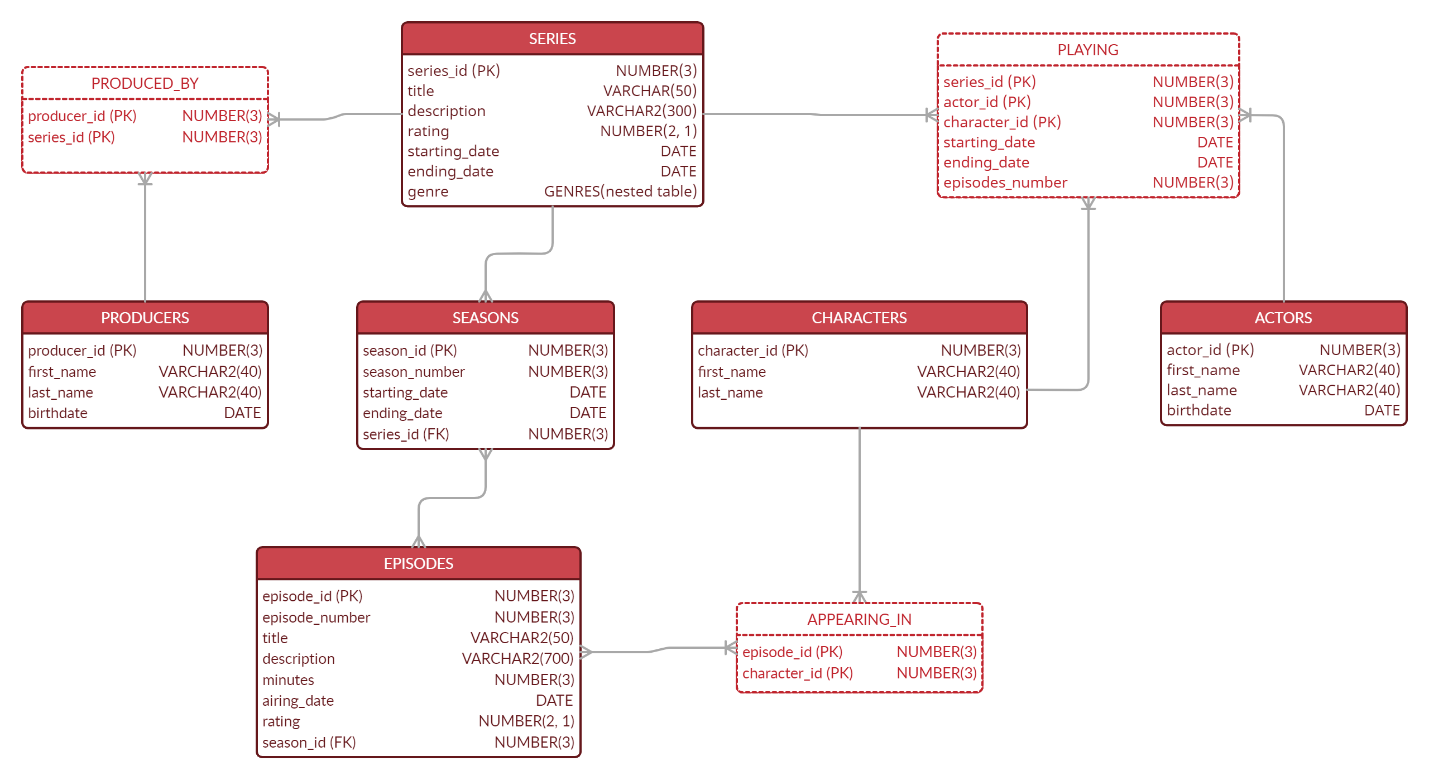
In acest sens, baza de date este conceputa cu 6 tabele independente (producers, series, seasons, episodes, actors, characters) si 3 tabele asociative pentru a rezolva legăturile de tip many to many (produced\_by, appearing\_in, playing). Din punct de vedere al relațiilor, sunt implementate următoarele reguli:

* un producător poate produce unul sau mai multe seriale, iar un serial poate fi produs de unul sau mai mulți producători;
* un actor joacă în unul sau mai multe seriale, un serial având unul sau mai mulți actori;
* un serial are unul sau mai multe sezoane, iar un sezon aparține unui singur serial;
* un sezon are unul sau mai multe episoade, iar un episod aparține unui singur sezon;
* un actor joacă unul sau mai multe personaje, iar un personaj este jucat de un singur actor;
* un personaj apare în unul sau mai multe episoade, în timp ce într-un episod pot apărea unul sau mai multe personaje.

# Diagrama E/R



# Diagrama conceptuala



# Implementarea in Oracle a diagramei conceptuale

1. Tabela series

Am ales să stochez categoriile unui serial sub forma de tablou imbricat.

CREATE OR REPLACE TYPE genres IS TABLE OF VARCHAR2(20);

/

CREATE TABLE series

(series\_id NUMBER(3) PRIMARY KEY,

title VARCHAR2(50) NOT NULL,

description VARCHAR(300),

rating NUMBER(2,1),

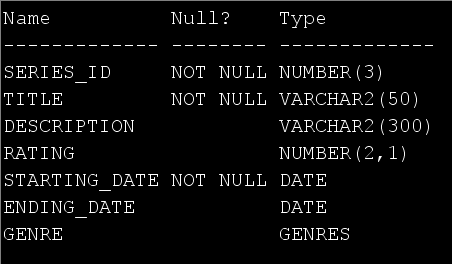
starting\_date DATE NOT NULL,

ending\_date DATE,

genre genres,

CONSTRAINT date\_series CHECK (starting\_date <= ending\_date))

NESTED TABLE genre STORE AS genre;



1. Tabela producers

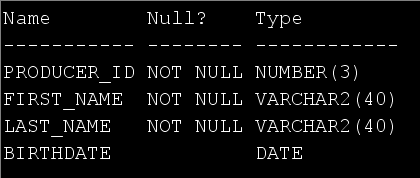
CREATE TABLE producers

(producer\_id NUMBER(3) PRIMARY KEY,

first\_name VARCHAR2(40) NOT NULL,

last\_name VARCHAR2(40) NOT NULL,

birthdate DATE);



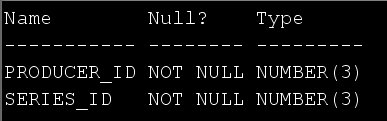
1. Tabela produced\_by

CREATE TABLE produced\_by

(producer\_id NUMBER(3) REFERENCES producers(producer\_id) ON DELETE CASCADE,

series\_id NUMBER(3) REFERENCES series(series\_id) ON DELETE CASCADE,

PRIMARY KEY(producer\_id, series\_id));



1. Tabela seasons

CREATE TABLE seasons

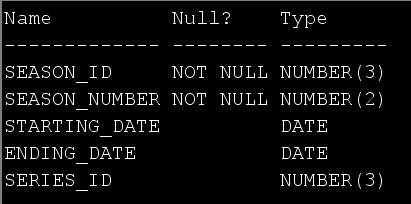
(season\_id NUMBER(3) PRIMARY KEY,

season\_number NUMBER(2) NOT NULL,

starting\_date DATE,

ending\_date DATE,

series\_id NUMBER(3) REFERENCES series(series\_id) ON DELETE CASCADE);



1. Tabela episodes

CREATE TABLE episodes

(episode\_id NUMBER(3) PRIMARY KEY,

episode\_number NUMBER(3) NOT NULL,

title VARCHAR2(50) NOT NULL,

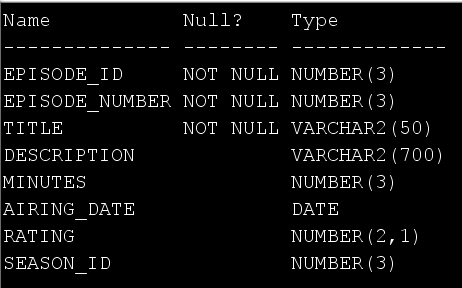
description VARCHAR2 (700),

minutes NUMBER(3),

airing\_date DATE,

rating NUMBER(2,1),

season\_id NUMBER(3) REFERENCES seasons(season\_id) ON DELETE CASCADE);



1. Tabela actors

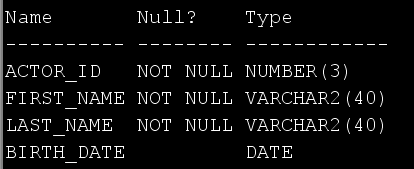
CREATE TABLE actors

(actor\_id NUMBER(3) PRIMARY KEY,

first\_name VARCHAR2(40) NOT NULL,

last\_name VARCHAR2(40) NOT NULL,

birth\_date DATE);



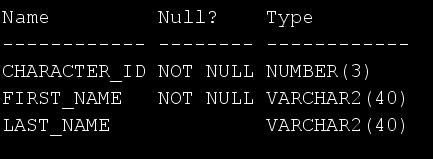
1. Tabela characters

CREATE TABLE characters

(character\_id NUMBER(3) PRIMARY KEY,

first\_name VARCHAR2(40) NOT NULL,

last\_name VARCHAR2(40));



1. Tabela playing

CREATE TABLE playing

(series\_id NUMBER(3) REFERENCES series(series\_id) ON DELETE CASCADE,

actor\_id NUMBER(3) REFERENCES actors(actor\_id) ON DELETE CASCADE,

character\_id NUMBER(3) REFERENCES characters(character\_id) ON DELETE CASCADE,

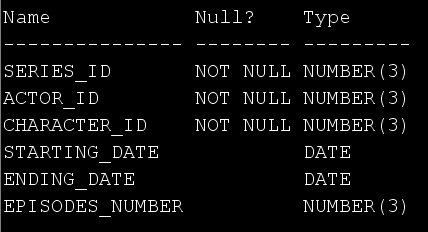
starting\_date DATE,

ending\_date DATE,

episodes\_number NUMBER(3),

PRIMARY KEY(series\_id, character\_id, actor\_id),

CONSTRAINT date\_playing CHECK (starting\_date <= ending\_date));



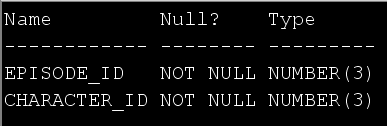
1. Tabela appearing\_in

CREATE TABLE appearing\_in

(episode\_id NUMBER(3) REFERENCES episodes(episode\_id) ON DELETE CASCADE,

character\_id NUMBER(3) REFERENCES characters(character\_id) ON DELETE CASCADE,

PRIMARY KEY(episode\_id, character\_id));



# Adăugarea de informații în tabele

1. Tabela series

INSERT INTO series

VALUES (1, 'Supernatural',

'Two brothers follow their father''s footsteps as hunters, fighting evil supernatural beings of many kinds, including monsters, demons and gods that roam the earth.',

8.4, TO\_DATE('13-Sep-2005', 'DD MONTH YYYY'), TO\_DATE('19-Nov-2020', 'DD MONTH YYYY'), genres('Drama', 'Fantasy', 'Horror'));

INSERT INTO series

VALUES (2, 'Gossip Girl',

'Privileged teens living on the Upper East Side of New York can hide no secret from the ruthless blogger who is always watching.',

7.4, TO\_DATE('19-Sep-2007', 'DD MONTH YYYY'), TO\_DATE('17-Dec-2012', 'DD MONTH YYYY'), genres('Drama', 'Romance'));

INSERT INTO series

VALUES (3, 'The Originals',

'A family of power-hungry thousand-year-old vampires look to take back the city that they built and dominate all those who have done them wrong.',

8.2, TO\_DATE('3-Oct-2013', 'DD MONTH YYYY'), TO\_DATE('1-Aug-2018', 'DD MONTH YYYY'), genres('Drama', 'Fantasy', 'Horror'));

INSERT INTO series

VALUES (4, 'Arrow',

'Spoiled billionaire playboy Oliver Queen is missing and presumed dead when his yacht is lost at sea. He returns five years later a changed man, determined to clean up the city as a hooded vigilante armed with a bow.',

7.5, TO\_DATE('10-Oct-2012', 'DD MONTH YYYY'), TO\_DATE('28-Jan-2020', 'DD MONTH YYYY'), genres('Action', 'Adventure', 'Crime'));

INSERT INTO series

VALUES (5, 'The 100',

'Set ninety-seven years after a nuclear war has destroyed civilization, when a spaceship housing humanity''s lone survivors sends one hundred juvenile delinquents back to Earth, in hopes of possibly re-populating the planet.',

7.6, TO\_DATE('19-Mar-2014', 'DD MONTH YYYY'), TO\_DATE('30-Sep-2020', 'DD MONTH YYYY'), genres('Drama', 'Mystery', 'Sci-Fi'));

INSERT INTO series

VALUES (6, 'Lucifer',

'Lucifer Morningstar has decided he''s had enough of being the dutiful servant in Hell and decides to spend some time on Earth to better understand humanity. He settles in Los Angeles - the City of Angels.',

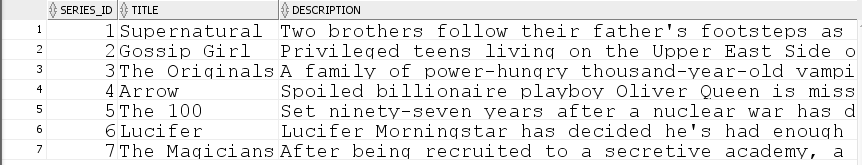
8.2, TO\_DATE('25-Jan-2016', 'DD MONTH YYYY'), NULL, genres('Crime', 'Drama', 'Fantasy'));

INSERT INTO series

VALUES (7, 'The Magicians',

'After being recruited to a secretive academy, a group of students discover that the magic they read about as children is very real-and more dangerous than they ever imagined.',

7.6, TO\_DATE('26-Dec-2015', 'DD MONTH YYYY'), TO\_DATE('1-Apr-2020', 'DD MONTH YYYY'), genres('Drama', 'Fantasy', 'Mystery'));



1. Tabela producers

INSERT INTO producers

VALUES (1, 'Eric', 'Kripke', '24-APR-1974');

INSERT INTO producers

VALUES (2, 'Stephanie', 'Savage', TO\_DATE('1969', 'YYYY'));

INSERT INTO producers

VALUES (3, 'Josh', 'Schwartz', '06-AUG-1976');

INSERT INTO producers

VALUES (4, 'Julie', 'Plec', '26-MAY-1972');

INSERT INTO producers

VALUES (5, 'Greg', 'Berlanti', '24-MAY-1972');

INSERT INTO producers

VALUES (6, 'Marc', 'Guggenheim', '24-SEP-1970');

INSERT INTO producers

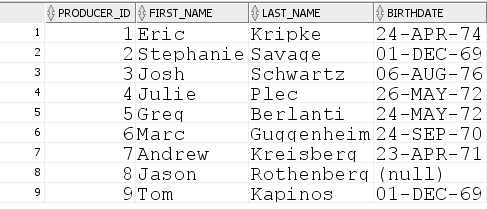
VALUES (7, 'Andrew', 'Kreisberg', '23-APR-1971');

INSERT INTO producers

VALUES (8, 'Jason', 'Rothenberg', null);

INSERT INTO producers

VALUES (9, 'Tom', 'Kapinos', TO\_DATE('1969', 'YYYY'));



1. Tabela produced\_by

INSERT INTO produced\_by

VALUES (1, 1);

INSERT INTO produced\_by

VALUES (2, 2);

INSERT INTO produced\_by

VALUES (3, 2);

INSERT INTO produced\_by

VALUES (4, 3);

INSERT INTO produced\_by

VALUES (5, 4);

INSERT INTO produced\_by

VALUES (6, 4);

INSERT INTO produced\_by

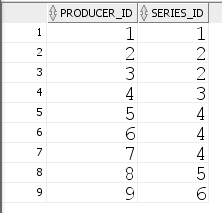
VALUES (7, 4);

INSERT INTO produced\_by

VALUES (8, 5);

INSERT INTO produced\_by

VALUES (9, 6);



1. Tabela seasons

INSERT INTO seasons

VALUES (1, 5, TO\_DATE('10-Sep-2009', 'DD MONTH YYYY'), TO\_DATE('13-May-2010', 'DD MONTH YYYY'), 1);

INSERT INTO seasons

VALUES (2, 15, TO\_DATE('10-Oct-2019', 'DD MONTH YYYY'), TO\_DATE('19-Nov-2020', 'DD MONTH YYYY'), 1);

INSERT INTO seasons

VALUES (3, 1, TO\_DATE('19-Sep-2007', 'DD MONTH YYYY'), TO\_DATE('19-May-2008', 'DD MONTH YYYY'), 2);

INSERT INTO seasons

VALUES (4, 6, TO\_DATE('8-Oct-2011', 'DD MONTH YYYY'), TO\_DATE('17-Dec-2012', 'DD MONTH YYYY'), 2);

INSERT INTO seasons

VALUES (5, 1, TO\_DATE('3-Oct-2013', 'DD MONTH YYYY'), TO\_DATE('13-May-2014', 'DD MONTH YYYY'), 3);

INSERT INTO seasons

VALUES (6, 3, TO\_DATE('8-Oct-2015', 'DD MONTH YYYY'), TO\_DATE('20-May-2016', 'DD MONTH YYYY'), 3);

INSERT INTO seasons

VALUES (7, 2, TO\_DATE('9-Oct-2013', 'DD MONTH YYYY'), TO\_DATE('14-May-2014', 'DD MONTH YYYY'), 4);

INSERT INTO seasons

VALUES (8, 3, TO\_DATE('8-Oct-2014', 'DD MONTH YYYY'), TO\_DATE('13-May-2015', 'DD MONTH YYYY'), 4);

INSERT INTO seasons

VALUES (9, 1, TO\_DATE('19-Mar-2014', 'DD MONTH YYYY'), TO\_DATE('11-Jun-2014', 'DD MONTH YYYY'), 5);

INSERT INTO seasons

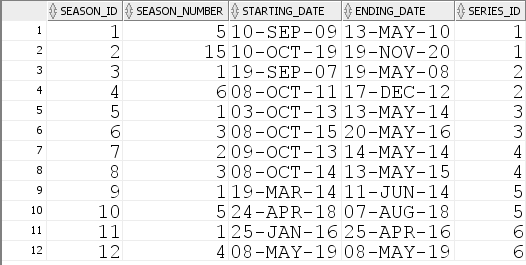
VALUES (10, 5, TO\_DATE('24-Apr-2018', 'DD MONTH YYYY'), TO\_DATE('7-Aug-2018', 'DD MONTH YYYY'), 5);

INSERT INTO seasons

VALUES (11, 1, TO\_DATE('25-Jan-2016', 'DD MONTH YYYY'), TO\_DATE('25-Apr-2016', 'DD MONTH YYYY'), 6);

INSERT INTO seasons

VALUES (12, 4, TO\_DATE('8-May-2019', 'DD MONTH YYYY'), TO\_DATE('8-May-2019', 'DD MONTH YYYY'), 6);



1. Tabela episodes

INSERT INTO episodes

VALUES (1, 22, 'Swan Song', 'With the Apocalypse looming, Sam and Dean realize they are out of options and make heart-breaking decisions that will change their lives forever.',

43, TO\_DATE('13-May-2010', 'DD MONTH YYYY'), 9.7, 1);

INSERT INTO episodes

VALUES (2, 8, 'Changing Channels', 'Sam and Dean catch up with the Trickster, who sends them through a dizzying montage of TV show parodies, inviting them to play along with their "roles" or be stuck in "TV Land" forever. But once Castiel shows up, the boys get an idea as to what the Trickster might be hiding and eventually come up with a surprising answer.',

43, TO\_DATE('5-Nov-2009', 'DD MONTH YYYY'), 9.7, 1);

INSERT INTO episodes

VALUES (3, 19, 'Inherit the Earth', 'Everything is on the line as the battle against God continues; a familiar face returns to join the fight.',

43, TO\_DATE('12-Nov-2020', 'DD MONTH YYYY'), 8.2, 2);

INSERT INTO episodes

VALUES (4, 10, 'New York, I Love You XOXO', 'In a fashionable farewell to remember, our favorite Upper East Siders join forces for one last soiree; The identity of Gossip Girl is finally revealed.',

41, TO\_DATE('17-Dec-2012', 'DD MONTH YYYY'), 9.1, 4);

INSERT INTO episodes

VALUES (5, 7, 'Victor/Victrola', 'Serena and Dan finally acknowledge they are crazy about each other; Jenny discovers a secret about her parents; Blair is once again devastated by Nate''s actions.',

42, TO\_DATE('7-Nov-2007', 'DD MONTH YYYY'), 8.2, 3);

INSERT INTO episodes

VALUES (6, 22, 'From a Cradle to a Grave', 'As the baby''s due date draws near, Klaus and Elijah embark on a search for Hayley, while Hayley determines to do whatever it takes to keep her unborn baby safe and away from the witches. Francesca takes a meeting with Oliver and Jackson to determine the future of the werewolves in New Orleans. In the aftermath of a surprising attack on Marcel and his vampires at the compound, Davina and Cami join resources to take down Klaus. Finally, in a desperate move to protect those most important to him, Klaus makes a heartbreaking decision.',

42, TO\_DATE('13-May-2014', 'DD MONTH YYYY'), 9.6, 5);

INSERT INTO episodes

VALUES (7, 22, 'The Bloody Crown', 'After months of thwarting off dangerous threats and deadly attacks, the Mikaelson siblings finally come face to face with the one person that could lead to their ultimate demise. With the stakes higher than ever and the compound overrun by an army of his oldest sworn enemies, Klaus is put on trial for centuries of atrocities he''s committed. Meanwhile, Marcel, who has been spiraling out of control following an act of betrayal by those closest to him, is stunned by the unexpected arrival of someone from his past. Finally, Elijah, Freya and Kol frantically search for a way to save their family before it''s too late. Hayley also appears.',

42, TO\_DATE('20-May-2016', 'DD MONTH YYYY'), 9.6, 6);

INSERT INTO episodes

VALUES (8, 14, 'A Streetcar Named Desire', 'The unexpected arrival of Stefan Salvatore may be the key to helping Freya rescue Klaus and Elijah from a magical trap.',

42, TO\_DATE('26-Feb-2016', 'DD MONTH YYYY'), 9.4, 6);

INSERT INTO episodes

VALUES (9, 9, 'The Climb', 'The League of Assassins give Oliver 48 hours to find Sara''s killer, or Starling City citizens will die. Oliver then has an epic confrontation with Ra''s al Ghul.',

42, TO\_DATE('10-Dec-2014', 'DD MONTH YYYY'), 9.6, 8);

INSERT INTO episodes

VALUES (10, 23, 'Unthinkable', 'As Oliver''s face off with Slade escalates, his resolve to the no-kill rule is tested. Especially as Slade targets the woman Oliver loves.',

44, TO\_DATE('14-May-2014', 'DD MONTH YYYY'), 9.5, 7);

INSERT INTO episodes

VALUES (11, 18, 'Deathstroke', 'After taking a ride home from Slade, Thea becomes his prisoner. Can Oliver and his friends save her? Also, someone close to Oliver is working for Slade, since his return from the Island after Tommy''s death.',

42, TO\_DATE('2-Apr-2014', 'DD MONTH YYYY'), 9.3, 7);

INSERT INTO episodes

VALUES (12, 13, 'Damocles: Part Two', 'Clarke and her friends must risk everything to fight one last battle for survival, only to glimpse an even darker threat to the last living valley on earth.',

42, TO\_DATE('7-Aug-2018', 'DD MONTH YYYY'), 9.6, 10);

INSERT INTO episodes

VALUES (13, 12, 'Damocles: Part One', 'In part one of the fifth season finale, Octavia leads her people into war. While behind enemy lines, our heroes must overcome their differences to save Wonkru from extinction.',

42, TO\_DATE('31-Jul-2018', 'DD MONTH YYYY'), 9.0, 10);

INSERT INTO episodes

VALUES (14, 12, 'We Are Grounders: Part 2', 'As the remaining members of the 100 face off against the Grounders, Jaha makes a noble sacrifice to ensure the Ark makes it to Earth.',

42, TO\_DATE('11-Jul-2018', 'DD MONTH YYYY'), 8.9, 9);

INSERT INTO episodes

VALUES (15, 11, 'We Are Grounders: Part 1', 'Clarke and Finn come face to face with a new enemy after Lincoln rescues them from Anya, while Bellamy, Raven, Octavia and Jasper deal with a vengeful Murphy. On the Ark, Jaha makes a plan to try and get to Earth.',

43, TO\_DATE('4-Jul-2018', 'DD MONTH YYYY'), 8.4, 9);

INSERT INTO episodes

VALUES (16, 10, 'Who''s da New King of Hell?', 'With murderous demons on the loose in Los Angeles, it''s up to Lucifer to rein in the chaos and protect the ones he most cares about.',

55, TO\_DATE('8-May-2019', 'DD MONTH YYYY'), 9.7, 12);

INSERT INTO episodes

VALUES (17, 7, 'Devil Is as Devil Does', 'Eve takes a more active role in her main man''s professional life. Meanwhile, Lucifer gets back to basics and Amenadiel fights for his family.',

47, TO\_DATE('8-May-2019', 'DD MONTH YYYY'), 9.3, 12);

INSERT INTO episodes

VALUES (18, 13, 'Take Me Back to Hell', 'When Lucifer is framed for murder, he and Chloe must work together to clear his name and prove the identity of the true killer.',

43, TO\_DATE('25-Apr-2016', 'DD MONTH YYYY'), 9.2, 11);



1. Tabela actors

INSERT INTO actors

VALUES (1, 'Jared', 'Padalecki', '19-JUL-1982');

INSERT INTO actors

VALUES (2, 'Jensen', 'Ackles', '01-MAR-1978');

INSERT INTO actors

VALUES (3, 'Misha', 'Collins', '20-AUG-1974');

INSERT INTO actors

VALUES (4, 'Mark', 'Sheppard', '30-MAY-1964');

INSERT INTO actors

VALUES (5, 'Alexander', 'Calvert', '15-JUL-1990');

INSERT INTO actors

VALUES (6, 'Rob', 'Benedict', '21-SEP-1970');

INSERT INTO actors

VALUES (7, 'Blake', 'Lively', '25-AUG-1987');

INSERT INTO actors

VALUES (8, 'Leighton', 'Master', '09-APR-1986');

INSERT INTO actors

VALUES (9, 'Penn', 'Badgley', '01-NOV-1986');

INSERT INTO actors

VALUES (10, 'Ed', 'Westwick', '27-JUN-1987');

INSERT INTO actors

VALUES (11, 'Chace', 'Crawford', '18-JUL-1985');

INSERT INTO actors

VALUES (12, 'Joseph', 'Morgan', '16-MAY-1981');

INSERT INTO actors

VALUES (13, 'Daniel', 'Gilles', '14-MAR-1976');

INSERT INTO actors

VALUES (14, 'Claire', 'Holt', '11-JUN-1988');

INSERT INTO actors

VALUES (15, 'Riley', 'Voelkel', '26-APR-1990');

INSERT INTO actors

VALUES (16, 'Nathaniel', 'Buzolic', '04-AUG-1983');

INSERT INTO actors

VALUES (17, 'Danielle Rose', 'Russell', '31-OCT-1999');

INSERT INTO actors

VALUES (18, 'Phoebe', 'Tonkin', '12-JUL-1989');

INSERT INTO actors

VALUES (19, 'Charles Michael', 'Davis', '01-DEC-1984');

INSERT INTO actors

VALUES (20, 'Stephen', 'Amell', '08-MAY-1981');

INSERT INTO actors

VALUES (21, 'Willa', 'Holland', '18-JUN-1991');

INSERT INTO actors

VALUES (22, 'Emily Bett', 'Rickards', '24-JUL-1991');

INSERT INTO actors

VALUES (23, 'David', 'Ramsey', '17-NOV-1971');

INSERT INTO actors

VALUES (24, 'Katie', 'Cassidy', '25-NOV-1986');

INSERT INTO actors

VALUES (25, 'Manu', 'Bennett', '10-OCT-1969');

INSERT INTO actors

VALUES (26, 'Eliza', 'Taylor', '24-OCT-1989');

INSERT INTO actors

VALUES (27, 'Marie', 'Avgeropoulos', '17-JUN-1986');

INSERT INTO actors

VALUES (28, 'Bob', 'Morley', '20-DEC-1984');

INSERT INTO actors

VALUES (29, 'Lindsey', 'Morgan', '27-FEB-1990');

INSERT INTO actors

VALUES (30, 'Richard', 'Harmon', '18-AUG-1991');

INSERT INTO actors

VALUES (31, 'Cristopher', 'Larkin', '02-OCT-1987');

INSERT INTO actors

VALUES (32, 'Tom', 'Ellis', '17-NOV-1978');

INSERT INTO actors

VALUES (33, 'Lauren', 'German', '29-NOV-1978');

INSERT INTO actors

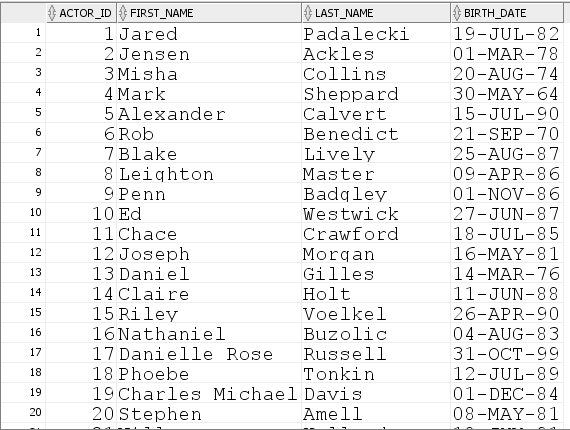
VALUES (34, 'David Bryan', 'Woodside', '25-JUL-1969');

INSERT INTO actors

VALUES (35, 'Lesley-Ann', 'Brandt', '02-DEC-1981');

INSERT INTO actors

VALUES (36, 'Rachel', 'Harris', '12-JAN-1968');



1. Tabela characters

INSERT INTO characters

VALUES (1, 'Sam', 'Winchester', 1);

INSERT INTO characters

VALUES (2, 'Dean', 'Winchester', 1);

INSERT INTO characters

VALUES (3, 'Castiel', null, 1);

INSERT INTO characters

VALUES (4, 'Crowley', null, 1);

INSERT INTO characters

VALUES (5, 'Jack', null, 1);

INSERT INTO characters

VALUES (6, 'God', null, 1);

INSERT INTO characters

VALUES (7, 'Serena', 'van der Woodsen' , 2);

INSERT INTO characters

VALUES (8, 'Blair', 'Waldorf', 2);

INSERT INTO characters

VALUES (9, 'Dan', 'Humphrey', 2);

INSERT INTO characters

VALUES (10, 'Chuck', 'Bass', 2);

INSERT INTO characters

VALUES (11, 'Nate', 'Archibald', 2);

INSERT INTO characters

VALUES (12, 'Klaus', 'Mikaelson', 3);

INSERT INTO characters

VALUES (13, 'Elijah', 'Mikaelson', 3);

INSERT INTO characters

VALUES (14, 'Rebekah', 'Mikaelson', 3);

INSERT INTO characters

VALUES (15, 'Freya', 'Mikaelson', 3);

INSERT INTO characters

VALUES (16, 'Kol', 'Mikaelson', 3);

INSERT INTO characters

VALUES (17, 'Hope', 'Mikaelson', 3);

INSERT INTO characters

VALUES (18, 'Hayley', 'Marshall', 3);

INSERT INTO characters

VALUES (19, 'Marcel', 'Gerard', 3);

INSERT INTO characters

VALUES (20, 'Oliver', 'Queen', 4);

INSERT INTO characters

VALUES (21, 'Thea', 'Queen', 4);

INSERT INTO characters

VALUES (22, 'Felicity', 'Smoak', 4);

INSERT INTO characters

VALUES (23, 'John', 'Diggle', 4);

INSERT INTO characters

VALUES (24, 'Laurel', 'Lance', 4);

INSERT INTO characters

VALUES (25, 'Slade', 'Wilson', 4);

INSERT INTO characters

VALUES (26, 'Clark', 'Griffin', 5);

INSERT INTO characters

VALUES (27, 'Octavia', 'Blake', 5);

INSERT INTO characters

VALUES (28, 'Bellamy', 'Blake', 5);

INSERT INTO characters

VALUES (29, 'Raven', 'Reyes', 5);

INSERT INTO characters

VALUES (30, 'John', 'Murphy', 5);

INSERT INTO characters

VALUES (31, 'Monty', 'Green', 5);

INSERT INTO characters

VALUES (32, 'Lucifer', 'Morningstar', 6);

INSERT INTO characters

VALUES (33, 'Chloe', 'Decker', 6);

INSERT INTO characters

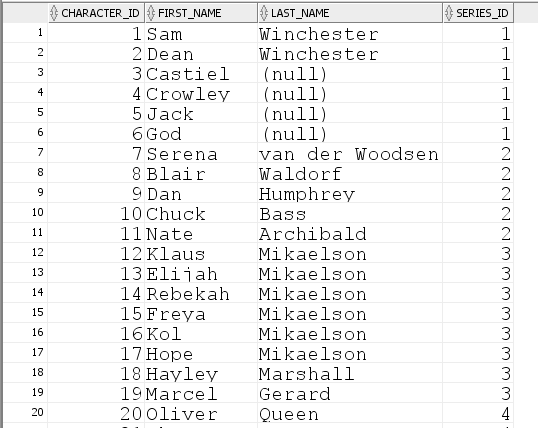
VALUES (34, 'Amenadiel', null, 6);

INSERT INTO characters

VALUES (35, 'Mazikeen',null, 6);

INSERT INTO characters

VALUES (36, 'Linda', 'Martin', 6);



1. Tabela playing

INSERT INTO playing

VALUES (1, 1, 1, TO\_DATE('2005', 'YYYY'), TO\_DATE('2020', 'YYYY'), 327);

INSERT INTO playing

VALUES (1, 2, 2, TO\_DATE('2005', 'YYYY'), TO\_DATE('2020', 'YYYY'), 327);

INSERT INTO playing

VALUES (1, 3, 3, TO\_DATE('2008', 'YYYY'), TO\_DATE('2020', 'YYYY'), 148);

INSERT INTO playing

VALUES (1, 4, 4, TO\_DATE('2009', 'YYYY'), TO\_DATE('2017', 'YYYY'), 72);

INSERT INTO playing

VALUES (1, 5, 5, TO\_DATE('2017', 'YYYY'), TO\_DATE('2020', 'YYYY'), 39);

INSERT INTO playing

VALUES (1, 6, 6, TO\_DATE('2009', 'YYYY'), TO\_DATE('2020', 'YYYY'), 9);

INSERT INTO playing

VALUES (2, 7, 7, TO\_DATE('2007', 'YYYY'), TO\_DATE('2012', 'YYYY'), 121);

INSERT INTO playing

VALUES (2, 8, 8, TO\_DATE('2007', 'YYYY'), TO\_DATE('2012', 'YYYY'), 121);

INSERT INTO playing

VALUES (2, 9, 9, TO\_DATE('2007', 'YYYY'), TO\_DATE('2012', 'YYYY'), 121);

INSERT INTO playing

VALUES (2, 10, 10, TO\_DATE('2007', 'YYYY'), TO\_DATE('2012', 'YYYY'), 121);

INSERT INTO playing

VALUES (2, 11, 11, TO\_DATE('2007', 'YYYY'), TO\_DATE('2012', 'YYYY'), 121);

INSERT INTO playing

VALUES (3, 12, 12, TO\_DATE('2013', 'YYYY'), TO\_DATE('2018', 'YYYY'), 92);

INSERT INTO playing

VALUES (3, 13, 13, TO\_DATE('2013', 'YYYY'), TO\_DATE('2018', 'YYYY'), 92);

INSERT INTO playing

VALUES (3, 14, 14, TO\_DATE('2013', 'YYYY'), TO\_DATE('2018', 'YYYY'), 40);

INSERT INTO playing

VALUES (3, 15, 15, TO\_DATE('2014', 'YYYY'), TO\_DATE('2018', 'YYYY'), 60);

INSERT INTO playing

VALUES (3, 16, 16, TO\_DATE('2013', 'YYYY'), TO\_DATE('2018', 'YYYY'), 24);

INSERT INTO playing

VALUES (3, 17, 17, TO\_DATE('2018', 'YYYY'), TO\_DATE('2018', 'YYYY'), 13);

INSERT INTO playing

VALUES (3, 18, 18, TO\_DATE('2013', 'YYYY'), TO\_DATE('2018', 'YYYY'), 86);

INSERT INTO playing

VALUES (3, 19, 19, TO\_DATE('2013', 'YYYY'), TO\_DATE('2018', 'YYYY'), 92);

INSERT INTO playing

VALUES (4, 20, 20, TO\_DATE('2012', 'YYYY'), TO\_DATE('2020', 'YYYY'), 170);

INSERT INTO playing

VALUES (4, 21, 21, TO\_DATE('2012', 'YYYY'), TO\_DATE('2020', 'YYYY'), 134);

INSERT INTO playing

VALUES (4, 22, 22, TO\_DATE('2012', 'YYYY'), TO\_DATE('2020', 'YYYY'), 157);

INSERT INTO playing

VALUES (4, 23, 23, TO\_DATE('2012', 'YYYY'), TO\_DATE('2020', 'YYYY'), 170);

INSERT INTO playing

VALUES (4, 24, 24, TO\_DATE('2012', 'YYYY'), TO\_DATE('2020', 'YYYY'), 153);

INSERT INTO playing

VALUES (4, 25, 25, TO\_DATE('2013', 'YYYY'), TO\_DATE('2017', 'YYYY'), 40);

INSERT INTO playing

VALUES (5, 26, 26, TO\_DATE('2014', 'YYYY'), TO\_DATE('2020', 'YYYY'), 100);

INSERT INTO playing

VALUES (5, 27, 27, TO\_DATE('2014', 'YYYY'), TO\_DATE('2020', 'YYYY'), 100);

INSERT INTO playing

VALUES (5, 28, 28, TO\_DATE('2014', 'YYYY'), TO\_DATE('2020', 'YYYY'), 97);

INSERT INTO playing

VALUES (5, 29, 29, TO\_DATE('2014', 'YYYY'), TO\_DATE('2020', 'YYYY'), 98);

INSERT INTO playing

VALUES (5, 30, 30, TO\_DATE('2014', 'YYYY'), TO\_DATE('2020', 'YYYY'), 90);

INSERT INTO playing

VALUES (5, 31, 31, TO\_DATE('2014', 'YYYY'), TO\_DATE('2019', 'YYYY'), 73);

INSERT INTO playing

VALUES (6, 32, 32, TO\_DATE('2016', 'YYYY'), null, 78);

INSERT INTO playing

VALUES (6, 33, 33, TO\_DATE('2016', 'YYYY'), null, 78);

INSERT INTO playing

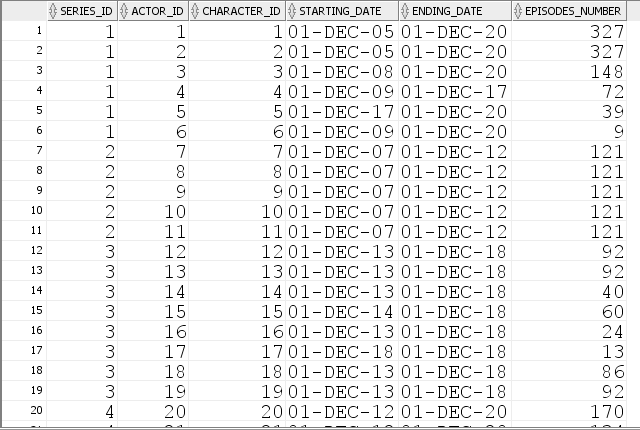
VALUES (6, 34, 34, TO\_DATE('2016', 'YYYY'), null, 77);

INSERT INTO playing

VALUES (6, 35, 35, TO\_DATE('2016', 'YYYY'), null, 77);

INSERT INTO playing

VALUES (6, 36, 36, TO\_DATE('2016', 'YYYY'), null, 77);



1. Tabela appearing\_in

INSERT INTO appearing\_in

VALUES (1, 1);

INSERT INTO appearing\_in

VALUES (1, 2);

INSERT INTO appearing\_in

VALUES (1, 3);

INSERT INTO appearing\_in

VALUES (2, 1);

INSERT INTO appearing\_in

VALUES (2, 2);

INSERT INTO appearing\_in

VALUES (2, 3);

INSERT INTO appearing\_in

VALUES (3, 1);

INSERT INTO appearing\_in

VALUES (3, 2);

INSERT INTO appearing\_in

VALUES (3, 3);

INSERT INTO appearing\_in

VALUES (3, 5);

INSERT INTO appearing\_in

VALUES (3, 6);

INSERT INTO appearing\_in

VALUES (4, 7);

INSERT INTO appearing\_in

VALUES (4, 8);

INSERT INTO appearing\_in

VALUES (4, 9);

INSERT INTO appearing\_in

VALUES (4, 10);

INSERT INTO appearing\_in

VALUES (4, 11);

INSERT INTO appearing\_in

VALUES (5, 7);

INSERT INTO appearing\_in

VALUES (5, 8);

INSERT INTO appearing\_in

VALUES (5, 9);

INSERT INTO appearing\_in

VALUES (5, 10);

INSERT INTO appearing\_in

VALUES (5, 11);

INSERT INTO appearing\_in

VALUES (6, 12);

INSERT INTO appearing\_in

VALUES (6, 13);

INSERT INTO appearing\_in

VALUES (6, 14);

INSERT INTO appearing\_in

VALUES (6, 18);

INSERT INTO appearing\_in

VALUES (6, 19);

INSERT INTO appearing\_in

VALUES (7, 12);

INSERT INTO appearing\_in

VALUES (7, 13);

INSERT INTO appearing\_in

VALUES (7, 14);

INSERT INTO appearing\_in

VALUES (7, 15);

INSERT INTO appearing\_in

VALUES (7, 16);

INSERT INTO appearing\_in

VALUES (7, 18);

INSERT INTO appearing\_in

VALUES (7, 19);

INSERT INTO appearing\_in

VALUES (8, 12);

INSERT INTO appearing\_in

VALUES (8, 13);

INSERT INTO appearing\_in

VALUES (8, 15);

INSERT INTO appearing\_in

VALUES (8, 16);

INSERT INTO appearing\_in

VALUES (8, 18);

INSERT INTO appearing\_in

VALUES (8, 19);

INSERT INTO appearing\_in

VALUES (9, 20);

INSERT INTO appearing\_in

VALUES (9, 21);

INSERT INTO appearing\_in

VALUES (9, 22);

INSERT INTO appearing\_in

VALUES (9, 23);

INSERT INTO appearing\_in

VALUES (9, 24);

INSERT INTO appearing\_in

VALUES (10, 20);

INSERT INTO appearing\_in

VALUES (10, 21);

INSERT INTO appearing\_in

VALUES (10, 22);

INSERT INTO appearing\_in

VALUES (10, 23);

INSERT INTO appearing\_in

VALUES (10, 24);

INSERT INTO appearing\_in

VALUES (10, 25);

INSERT INTO appearing\_in

VALUES (11, 20);

INSERT INTO appearing\_in

VALUES (11, 21);

INSERT INTO appearing\_in

VALUES (11, 22);

INSERT INTO appearing\_in

VALUES (11, 23);

INSERT INTO appearing\_in

VALUES (11, 24);

INSERT INTO appearing\_in

VALUES (11, 25);

INSERT INTO appearing\_in

VALUES (12, 26);

INSERT INTO appearing\_in

VALUES (12, 27);

INSERT INTO appearing\_in

VALUES (12, 28);

INSERT INTO appearing\_in

VALUES (12, 29);

INSERT INTO appearing\_in

VALUES (12, 30);

INSERT INTO appearing\_in

VALUES (12, 31);

INSERT INTO appearing\_in

VALUES (13, 26);

INSERT INTO appearing\_in

VALUES (13, 27);

INSERT INTO appearing\_in

VALUES (13, 28);

INSERT INTO appearing\_in

VALUES (13, 29);

INSERT INTO appearing\_in

VALUES (13, 30);

INSERT INTO appearing\_in

VALUES (13, 31);

INSERT INTO appearing\_in

VALUES (14, 26);

INSERT INTO appearing\_in

VALUES (14, 27);

INSERT INTO appearing\_in

VALUES (14, 28);

INSERT INTO appearing\_in

VALUES (14, 29);

INSERT INTO appearing\_in

VALUES (14, 30);

INSERT INTO appearing\_in

VALUES (14, 31);

INSERT INTO appearing\_in

VALUES (15, 26);

INSERT INTO appearing\_in

VALUES (15, 27);

INSERT INTO appearing\_in

VALUES (15, 28);

INSERT INTO appearing\_in

VALUES (15, 29);

INSERT INTO appearing\_in

VALUES (15, 30);

INSERT INTO appearing\_in

VALUES (15, 31);

INSERT INTO appearing\_in

VALUES (16, 32);

INSERT INTO appearing\_in

VALUES (16, 33);

INSERT INTO appearing\_in

VALUES (16, 34);

INSERT INTO appearing\_in

VALUES (16, 35);

INSERT INTO appearing\_in

VALUES (16, 36);

INSERT INTO appearing\_in

VALUES (17, 32);

INSERT INTO appearing\_in

VALUES (17, 33);

INSERT INTO appearing\_in

VALUES (17, 34);

INSERT INTO appearing\_in

VALUES (17, 35);

INSERT INTO appearing\_in

VALUES (17, 36);

INSERT INTO appearing\_in

VALUES (18, 32);

INSERT INTO appearing\_in

VALUES (18, 33);

INSERT INTO appearing\_in

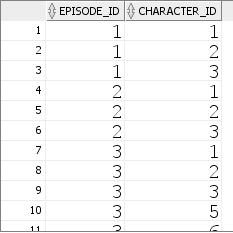
VALUES (18, 34);

INSERT INTO appearing\_in

VALUES (18, 35);

INSERT INTO appearing\_in

VALUES (18, 36);



# Definirea unui subprogram stocat care să utilizeze un tip de colecție studiat (tablou imbricat)

Sa se modifice lista de categorii a unui serial specificat:

i) adăugarea unei categorii (se da valoarea noii categori plus cuvântul 'INSERTING')

ii) ștergerea unei categorii (se da valoarea categoriei plus cuvântul 'DELETING')

iii) modificarea unei categorii (se da valoarea vechii categorii, noii categori plus cuvântul 'UPDATING')

CREATE OR REPLACE PROCEDURE modificare\_categorii

(serial series.title%TYPE,

categ1 VARCHAR2,

optiune VARCHAR2,

categ2 VARCHAR2 := NULL)

AS

categorii genres;

i INTEGER;

BEGIN

-- obtinere lista categorii pentru seraialul dat

SELECT genre INTO categorii

FROM series

WHERE title = INITCAP(serial);

IF UPPER(optiune) = 'INSERTING' THEN

IF categ1 IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Categoria nou introdusa nu poate sa fie NULL');

ELSE

-- adaugarea unei categorii noi

categorii.extend();

categorii(categorii.last) := INITCAP(categ1);

END IF;

ELSIF UPPER(optiune) = 'DELETING' THEN

IF categ1 IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Categoria de sters nu poate sa fie NULL');

ELSE

-- determinarea pozitiei categoriei ce trebuie stearsa

i := categorii.FIRST;

WHILE i <= categorii.LAST AND categorii(i) <> INITCAP(categ1) LOOP

i := categorii.NEXT(i);

END LOOP;

IF i IS NOT NULL THEN

-- stergerea categoriei

categorii.DELETE(i);

ELSE

RAISE\_APPLICATION\_ERROR(-20002, 'Nu exista categoria introdusa');

END IF;

END IF;

ELSIF UPPER(optiune) = 'UPDATING' THEN

IF categ1 IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Categoria de actualizat nu poate sa fie NULL');

ELSE

IF categ2 IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Categoria nou introdusa nu poate sa fie NULL');

ELSE

-- determinarea pozitiei categoriei ce trebuie modificata

i := categorii.FIRST;

WHILE i <= categorii.LAST AND categorii(i) <> INITCAP(categ1) LOOP

i := categorii.NEXT(i);

END LOOP;

IF i IS NOT NULL THEN

-- odificarea categoriei

categorii(i) := INITCAP(categ2);

ELSE

RAISE\_APPLICATION\_ERROR(-20002, 'Nu exista categoria introdusa');

END IF;

END IF;

END IF;

ELSE

RAISE\_APPLICATION\_ERROR(-20003, 'Optiunea introdusa este gresita');

END IF;

-- actualizare lista categorii

UPDATE series

SET genre = categorii

WHERE title = serial;

DBMS\_OUTPUT.PUT\_LINE('Lista de categorii a fost actualizata cu succes');

EXCEPTION

WHEN no\_data\_found THEN

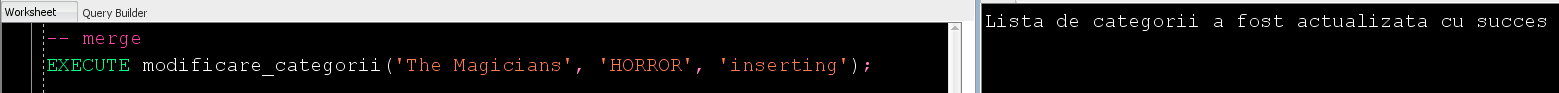
RAISE\_APPLICATION\_ERROR(-20004, 'Nu exista serial cu numele dat');

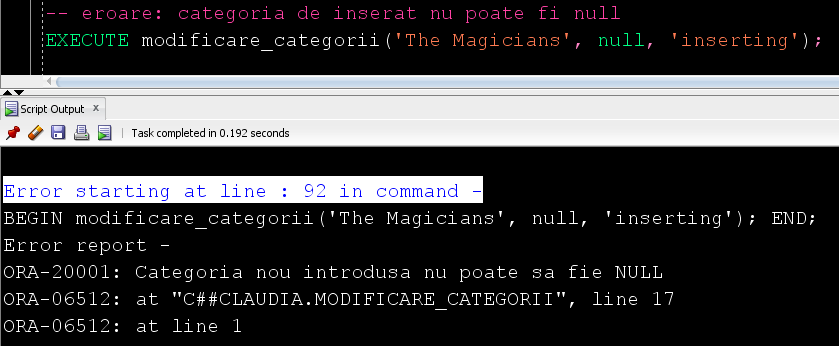
WHEN too\_many\_rows THEN

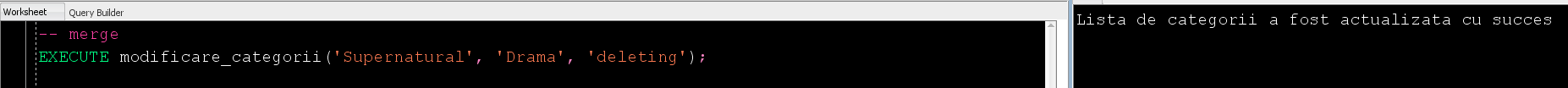
RAISE\_APPLICATION\_ERROR(-20005, 'Exista mai multe seriale cu acest nume');

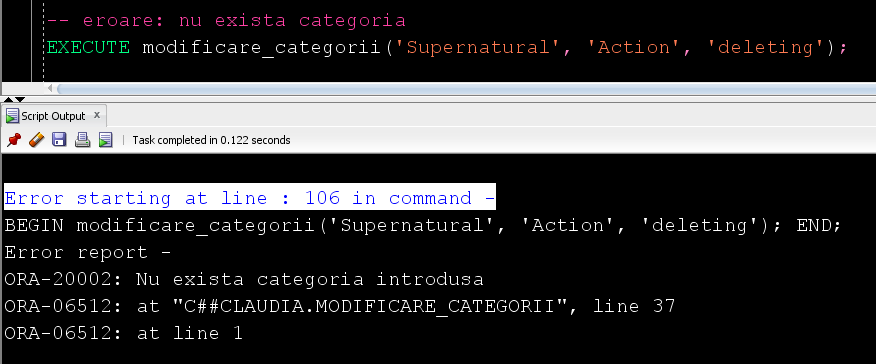
END;

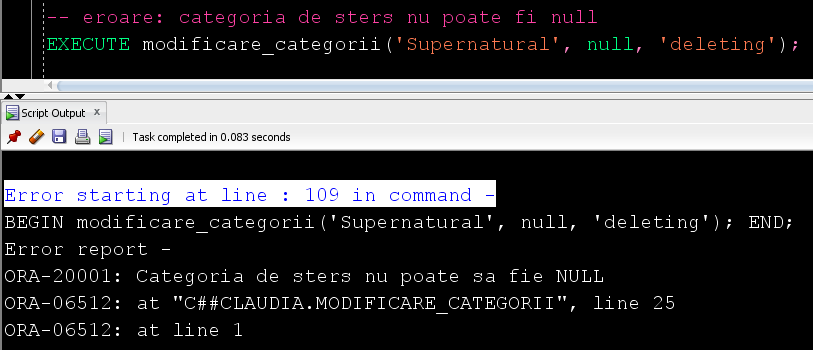
/

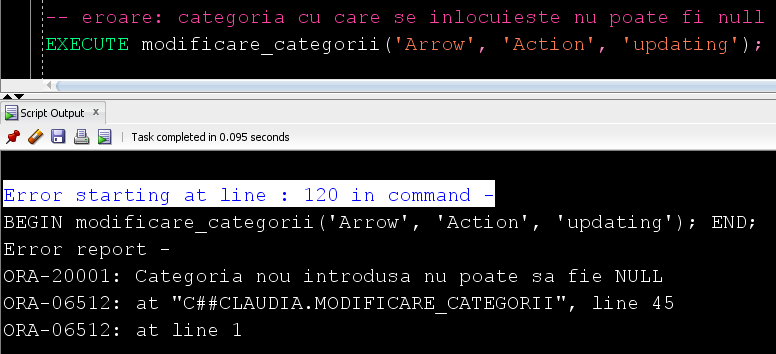


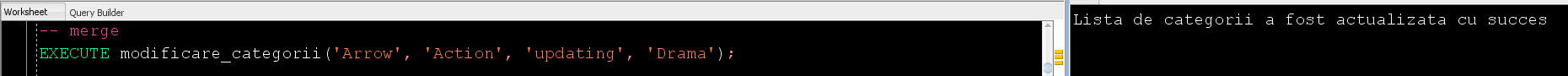


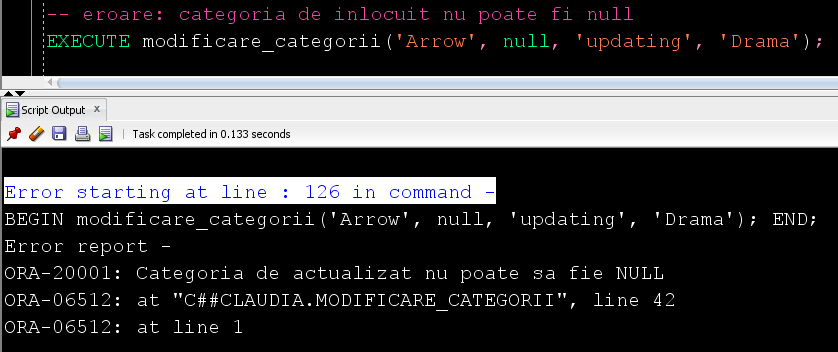


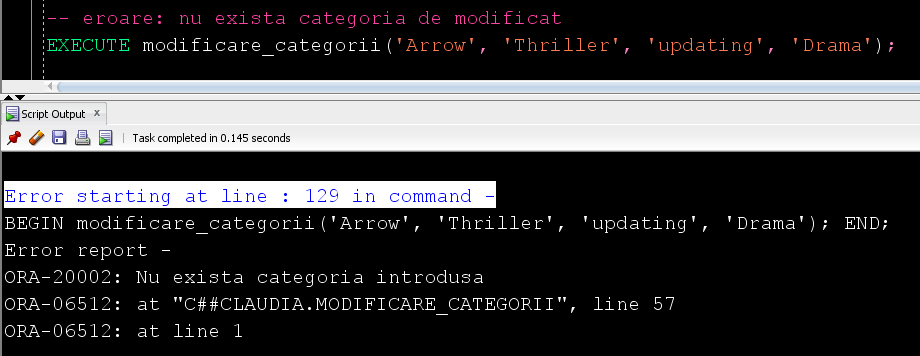


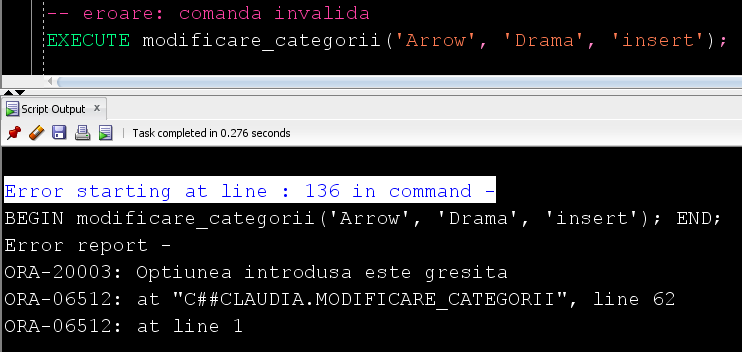












# Definirea unui subprogram stocat care să utilizeze un tip de cursor studiat (expresii cursor)

Sa se afișeze toate sezoanele si episoadele unui serial specificat (sezon: număr, data început, data sfârșit; episod: număr, nume, descriere, rating).

CREATE OR REPLACE PROCEDURE afisare\_episoade

(serial series.title%TYPE)

AS

TYPE ref\_cursor IS REF CURSOR;

CURSOR sezoane (id\_serial NUMBER) IS

SELECT season\_number, starting\_date, ending\_date,

CURSOR (SELECT episode\_number, title, description, rating

FROM episodes e

WHERE e.season\_id = s.season\_id)

FROM seasons s

WHERE series\_id = id\_serial;

episoade ref\_cursor;

id\_serial series.series\_id%TYPE;

numar\_sez seasons.season\_number%TYPE;

inceput\_sez seasons.starting\_date%TYPE;

sfarsit\_sez seasons.ending\_date%TYPE;

TYPE ep IS RECORD (numar episodes.episode\_number%TYPE,

titlu episodes.title%TYPE,

descriere episodes.description%TYPE,

rating episodes.rating%TYPE);

episod ep;

exista\_sezoane BOOLEAN := FALSE;

exista\_episoade BOOLEAN;

BEGIN

-- determinare id serial

SELECT series\_id INTO id\_serial

FROM series

WHERE title = serial;

OPEN sezoane(id\_serial);

LOOP

FETCH sezoane INTO numar\_sez, inceput\_sez, sfarsit\_sez, episoade;

EXIT WHEN sezoane%NOTFOUND;

exista\_sezoane := TRUE;

-- afisare sezon

DBMS\_OUTPUT.PUT\_LINE('------------------------------------------');

DBMS\_OUTPUT.PUT('SEZONUL ' || numar\_sez || ': ' || inceput\_sez || ' - ');

IF sfarsit\_sez IS NULL THEN

-- sezonul se afla in derulare

DBMS\_OUTPUT.PUT\_LINE('prezent');

ELSE

DBMS\_OUTPUT.PUT\_LINE(sfarsit\_sez);

END IF;

-- afisare episoade

exista\_episoade := FALSE;

LOOP

FETCH episoade INTO episod;

EXIT WHEN episoade%NOTFOUND;

exista\_episoade := TRUE;

DBMS\_OUTPUT.PUT\_LINE(episod.numar || '. ' || episod.titlu || ' - ' || episod.rating);

DBMS\_OUTPUT.PUT\_LINE('Synopsis: ' || episod.descriere);

DBMS\_OUTPUT.PUT\_LINE('');

END LOOP;

IF NOT exista\_episoade THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista episoade pentru acest sezon');

END IF;

DBMS\_OUTPUT.PUT\_LINE('------------------------------------------');

DBMS\_OUTPUT.PUT\_LINE('');

END LOOP;

CLOSE sezoane;

IF NOT exista\_sezoane THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista sezoane pentru acest serial');

END IF;

EXCEPTION

WHEN no\_data\_found THEN

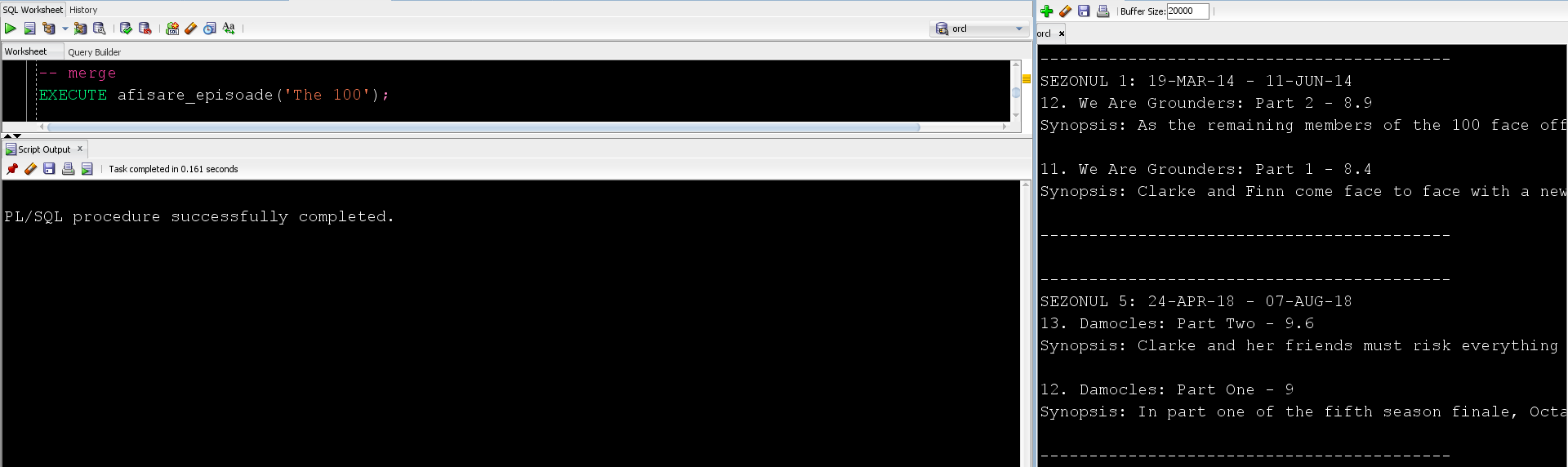
RAISE\_APPLICATION\_ERROR(-20004, 'Nu exista serial cu numele dat');

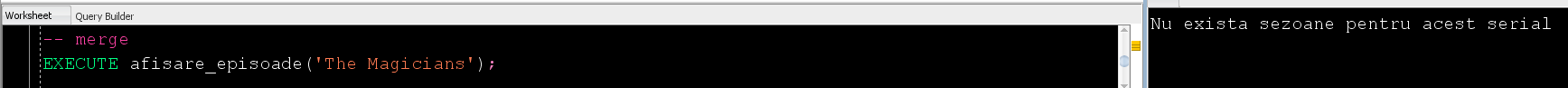
WHEN too\_many\_rows THEN

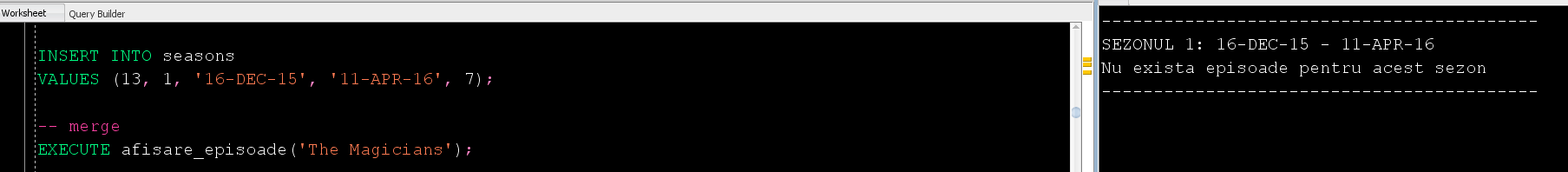
RAISE\_APPLICATION\_ERROR(-20005, 'Exista mai multe seriale cu acest nume');

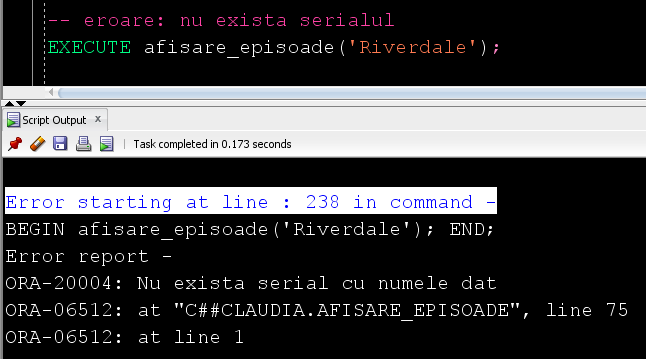
END;

/









# Definirea unei funcții care să utilizeze trei tabele diferite

Sa se determine numărul de episoade in care joaca un anumit actor într-o anumita perioada de timp.

CREATE OR REPLACE FUNCTION nr\_episoade

(prenume actors.first\_name%TYPE := NULL,

nume actors.last\_name%TYPE := NULL,

inceput DATE,

sfarsit DATE)

RETURN NUMBER IS

nr\_ep NUMBER(3);

id\_actor actors.actor\_id%TYPE;

BEGIN

IF inceput > sfarsit THEN

RAISE\_APPLICATION\_ERROR(-20006, 'Data de inceput trebuie sa fie mai mica decat data de sfarsit');

RETURN -1;

END IF;

IF prenume IS NULL AND nume IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20007, 'Nu poate sa fie si numele si prenumele NULL');

RETURN -1;

END IF;

-- determinarea id-ului actorului dat

-- (acest pas se face separat ca sa se poata arunca exceptie in cazul in care

-- nu exista actorulul sau exista mai multi acotri cu acest nume)

IF nume IS NOT NULL AND prenume IS NOT NULL THEN

-- numele si prenumele nusunt NULL

SELECT actor\_id INTO id\_actor

FROM actors

WHERE first\_name = prenume

AND last\_name = nume;

ELSIF nume IS NULL AND prenume IS NOT NULL THEN

-- prenumele nu este NULL

SELECT actor\_id INTO id\_actor

FROM actors

WHERE first\_name = prenume;

ELSE

-- numele nu este NULL

SELECT actor\_id INTO id\_actor

FROM actors

WHERE last\_name = nume;

END IF;

SELECT COUNT(\*) INTO nr\_ep

FROM playing p JOIN characters ch ON (p.character\_id = ch.character\_id)

JOIN appearing\_in ap ON (ch.character\_id = ap.character\_id)

JOIN episodes e ON (ap.episode\_id = e.episode\_id)

WHERE p.actor\_id = id\_actor

AND (p.starting\_date <= sfarsit AND p.ending\_date >= inceput)

AND e.airing\_date BETWEEN inceput AND sfarsit;

RETURN nr\_ep;

EXCEPTION

WHEN no\_data\_found THEN

RAISE\_APPLICATION\_ERROR(-20004, 'Nu exista actor cu numele dat');

RETURN -1;

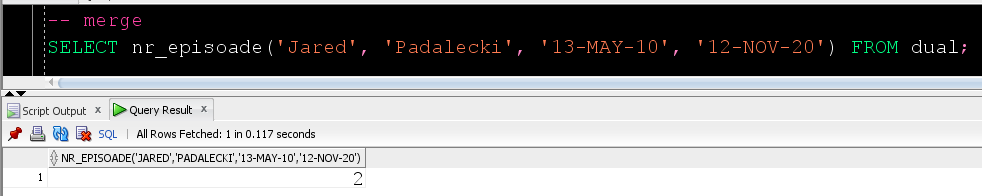
WHEN too\_many\_rows THEN

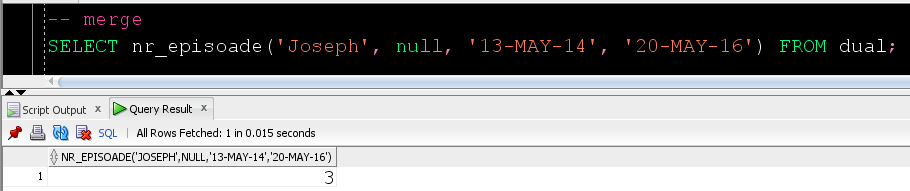
RAISE\_APPLICATION\_ERROR(-20005, 'Exista mai multi actori cu acest nume');

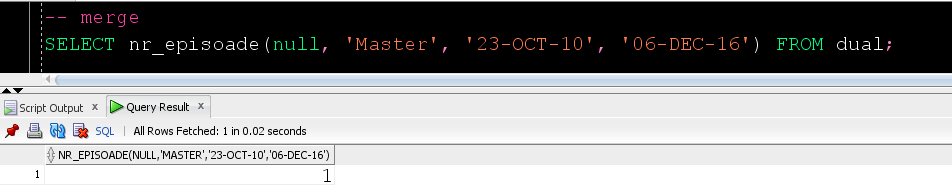
RETURN -1;

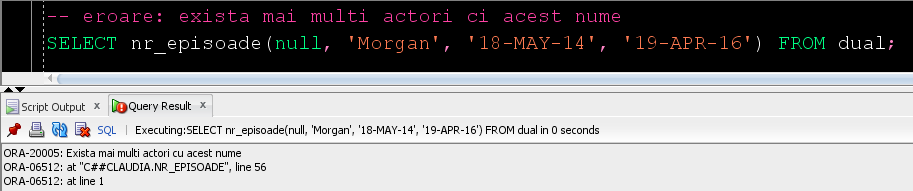
END;

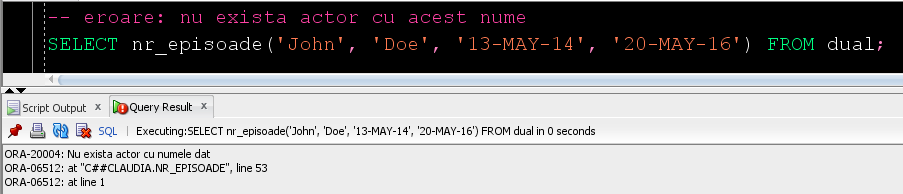
/

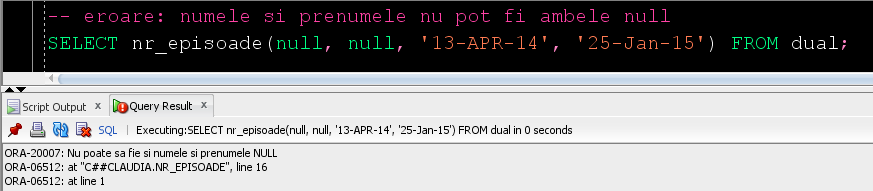


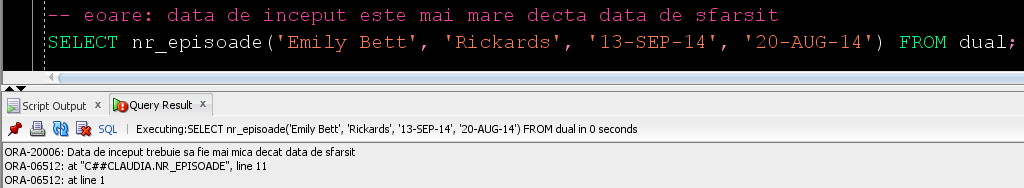












# Definirea unei proceduri care să utilizeze cinci tabele diferite

Sa se afișeze pentru un producător specificat fiecare serial pe care îl produce si lista actorilor împreună cu personajele pe care le interpretează in respectivul serial.

CREATE OR REPLACE PROCEDURE afisare\_seriale\_producator

(prenume producers.first\_name%TYPE DEFAULT NULL,

nume producers.last\_name%TYPE DEFAULT NULL)

AS

TYPE pers IS RECORD (prenume producers.first\_name%TYPE,

nume producers.last\_name%TYPE);

TYPE tab\_pers IS TABLE OF pers;

CURSOR actori (id\_serial NUMBER) IS

SELECT a.actor\_id, a.first\_name, a.last\_name

FROM actors a JOIN playing p ON(a.actor\_id = p.actor\_id)

WHERE series\_id = id\_serial;

personaje tab\_pers;

id\_producator producers.producer\_id%TYPE;

i INTEGER;

exista\_actori BOOLEAN;

BEGIN

IF prenume IS NULL AND nume IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20007, 'Nu poate sa fie si numele si prenumele NULL');

END IF;

-- determinarea id-ului producatorului dat

-- (acest pas se face separat ca sa se poata arunca exceptie in cazul in care

-- nu exista producatorul sau exista mai multi producatori cu acest nume)

IF nume IS NOT NULL AND prenume IS NOT NULL THEN

-- numele si prenumele nusunt NULL

SELECT producer\_id INTO id\_producator

FROM producers

WHERE first\_name = prenume

AND last\_name = nume;

ELSIF nume IS NULL AND prenume IS NOT NULL THEN

-- prenumele nu este NULL

SELECT producer\_id INTO id\_producator

FROM producers

WHERE first\_name = prenume;

ELSE

-- numele nu este NULL

SELECT producer\_id INTO id\_producator

FROM producers

WHERE last\_name = nume;

END IF;

FOR serial IN (SELECT series\_id, title

FROM produced\_by JOIN series USING (series\_id)

WHERE producer\_id = id\_producator)

LOOP

-- afisare serial

DBMS\_OUTPUT.PUT\_LINE('--- ' || UPPER(serial.title) || ' ---');

-- afisare actori

DBMS\_OUTPUT.PUT\_LINE('--- Actori: ');

exista\_actori := FALSE;

FOR actor in actori(serial.series\_id) LOOP

exista\_actori := TRUE;

DBMS\_OUTPUT.PUT(actor.first\_name || ' ' || actor.last\_name || ' - ');

-- afisare personaje jucate de actor

SELECT c.first\_name, c.last\_name BULK COLLECT INTO personaje

FROM characters c JOIN playing p USING(character\_id)

WHERE actor\_id = actor.actor\_id;

i := personaje.FIRST;

WHILE i<= personaje.LAST LOOP

DBMS\_OUTPUT.PUT(personaje(i).prenume || ' ' || personaje(i).nume);

IF i <> personaje.LAST THEN

DBMS\_OUTPUT.PUT(', ');

END IF;

i := personaje.NEXT(i);

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('');

END LOOP;

IF NOT exista\_actori THEN

-- nu exista actori

DBMS\_OUTPUT.PUT\_LINE('Nu exista actori');

END IF;

DBMS\_OUTPUT.PUT\_LINE('');

DBMS\_OUTPUT.PUT\_LINE('----------------------------------------------------------');

END LOOP;

EXCEPTION

WHEN no\_data\_found THEN

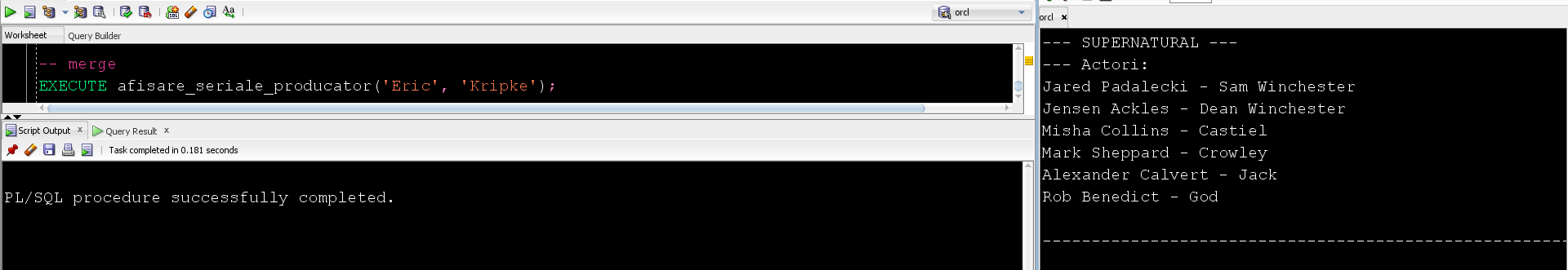
RAISE\_APPLICATION\_ERROR(-20004, 'Nu exista producator cu numele dat');

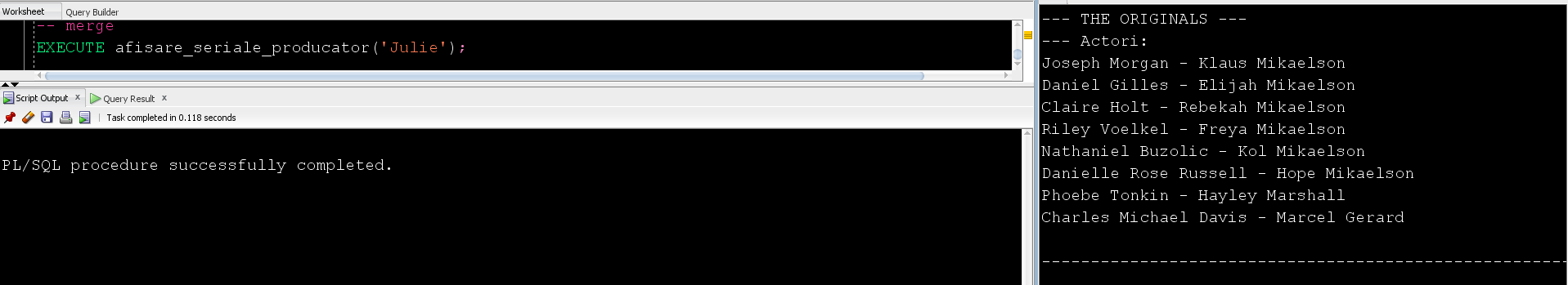
WHEN too\_many\_rows THEN

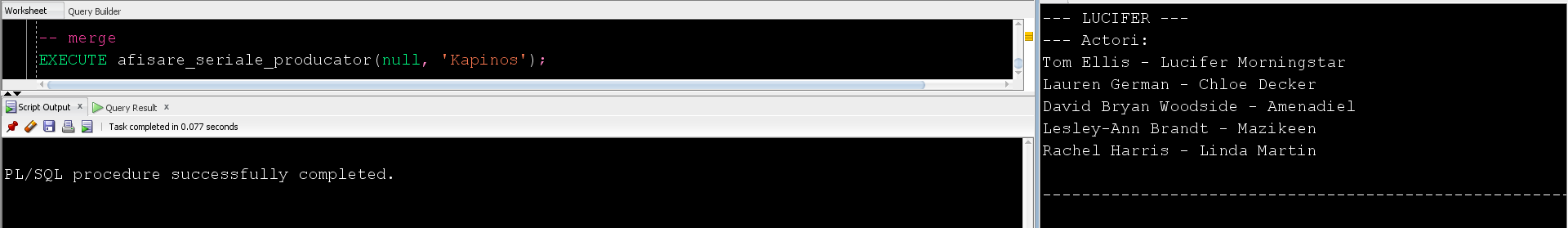
RAISE\_APPLICATION\_ERROR(-20005, 'Exista mai multi producatori cu acest nume');

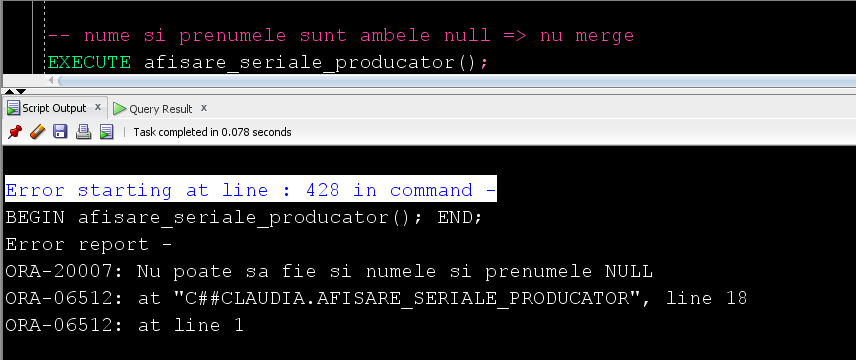
END;

/

****

****

****

****

# Definirea unui trigger LMD la nivel de comanda

Modificarea tabelei series nu se poate fi realizata decât de userul c##claudia, în intervalul Luni - Vineri intre orele 8 - 20.

CREATE OR REPLACE TRIGGER modificare\_serial

BEFORE INSERT OR UPDATE OR DELETE ON series

BEGIN

IF UPPER(SYS.LOGIN\_USER) <> 'C##CLAUDIA' THEN

RAISE\_APPLICATION\_ERROR(-20008, 'Nu aveti dreptul de a modifica acest tabel');

ELSIF TO\_CHAR(SYSTIMESTAMP, 'D') IN (1, 7) THEN

RAISE\_APPLICATION\_ERROR(-20009, 'Nu se poate modifica tabelul in zilele de weekend');

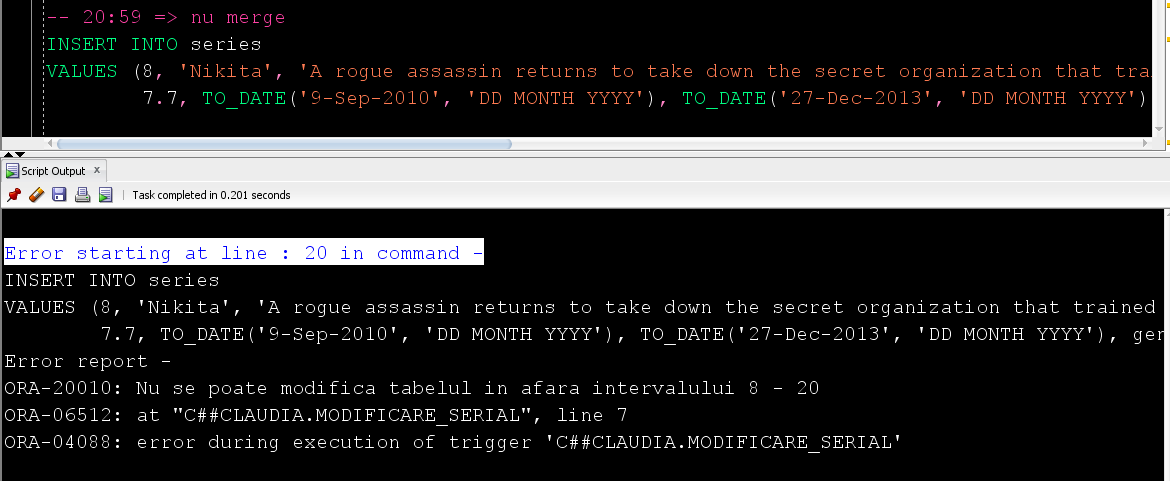
ELSIF TO\_CHAR(SYSTIMESTAMP,'HH24') NOT BETWEEN 8 AND 20 THEN

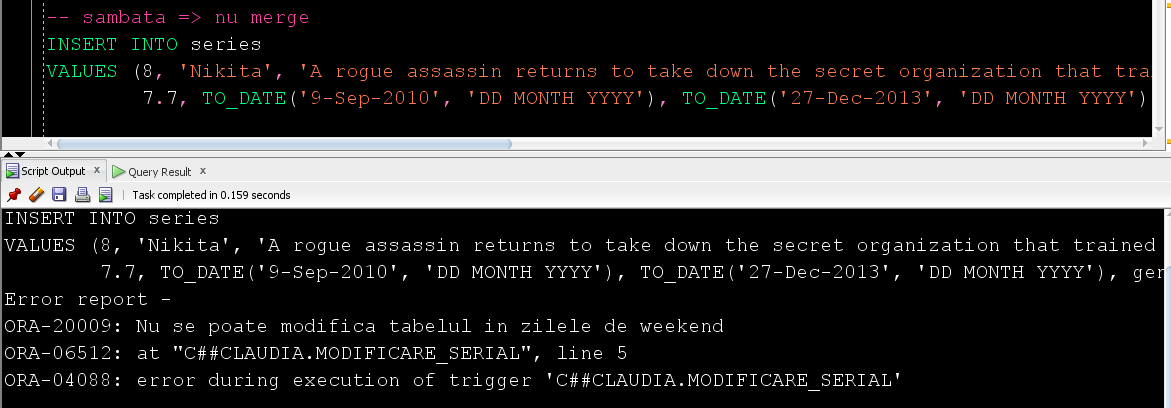
RAISE\_APPLICATION\_ERROR(-20010, 'Nu se poate modifica tabelul in afara intervalului 8 - 20');

END IF;

END;

/





# Definirea unui trigger LMD la nivel de linie

Un serial nu poate sa aibă mai mult de 4 episoade.

-- creare copie a tabelului episodes

CREATE TABLE episodes\_cpy

AS SELECT \* FROM episodes;

-- functie care returneaza nr de episoade ale unui sezon

CREATE OR REPLACE FUNCTION nr\_episoade\_serial

(id\_sez seasons.season\_id%TYPE)

RETURN NUMBER IS

nr\_ep NUMBER(1);

BEGIN

SELECT COUNT(\*) INTO nr\_ep

FROM episodes\_cpy JOIN seasons USING(season\_id)

JOIN series USING(series\_id)

WHERE series\_id = (SELECT series\_id

FROM seasons

WHERE season\_id = id\_sez);

RETURN nr\_ep;

END;

/

-- creare trigger

CREATE OR REPLACE TRIGGER modificare\_episodes

BEFORE INSERT OR UPDATE on episodes

FOR EACH ROW

BEGIN

IF nr\_episoade\_serial(:NEW.season\_id) = 4 THEN

RAISE\_APPLICATION\_ERROR(-20011, 'Un serial nu poate avea mai mult de 4 episoade');

END IF;

END;

/

-- trigger care actualizeaza tabela episodes\_cpy

CREATE OR REPLACE TRIGGER actualizare\_episodes\_cpy

AFTER INSERT OR UPDATE OR DELETE ON episodes

FOR EACH ROW

BEGIN

IF INSERTING THEN

INSERT INTO episodes\_cpy

VALUES (:NEW.episode\_id, :NEW.episode\_number, :NEW.title, :NEW.description,

:NEW.minutes, :NEW.airing\_date, :NEW.rating, :NEW.season\_id);

ELSIF UPDATING THEN

UPDATE episodes\_cpy

SET episode\_number = :NEW.episode\_number,

title = :NEW.title,

description = :NEW.description,

minutes = :NEW.minutes,

airing\_date = :NEW.airing\_date,

rating = :NEW.rating,

season\_id = :NEW.season\_id

WHERE episode\_id = :OLD.episode\_id;

ELSE

DELETE FROM episodes\_cpy

WHERE episode\_id = :OLD.episode\_id;

END IF;

END;

/

-- serialul are doar 3 episoade => merge

INSERT INTO episodes

VALUES (19, 19, 'No More Heartbreaks', 'Everyone joins together in an attempt to save Cami''s life.',

41, '29-APR-2016', 9.3, 6);

-- serialul are deja 4 episoade => nu merge

INSERT INTO episodes

VALUES (20, 11, 'Wild at Heart', 'Elijah learns that Aya might have knowlege about an elusive weapon that can take down Original Vampire for good. Davina receives a tempting offer which brings closer to reuniting Kol.',

42, '05-FEB-2016', 8.6, 6);

-- serialul are deja 4 episoade => nu merge

UPDATE episodes

SET season\_id = 6

WHERE episode\_id = 7;

-- serialul are doar 3 episoade, deci nu merg inserate alte 3 => nu merge

CREATE SEQUENCE sec\_episodes

START WITH 20

INCREMENT BY 1;

BEGIN

FOR i IN 1..5 LOOP

INSERT INTO episodes

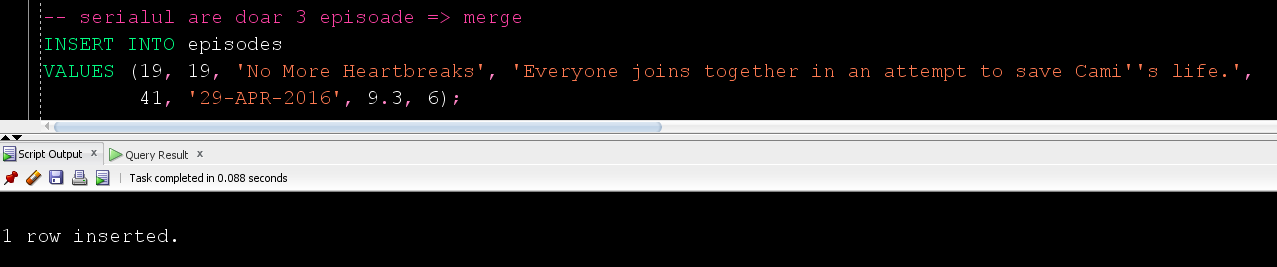
VALUES (sec\_episodes.NEXTVAL, 2, 'Sara', 'Team Arrow is in pursuit of a new villain who poses a threat to Starling City. Meanwhile, Oliver is worried about not having heard from Thea.',

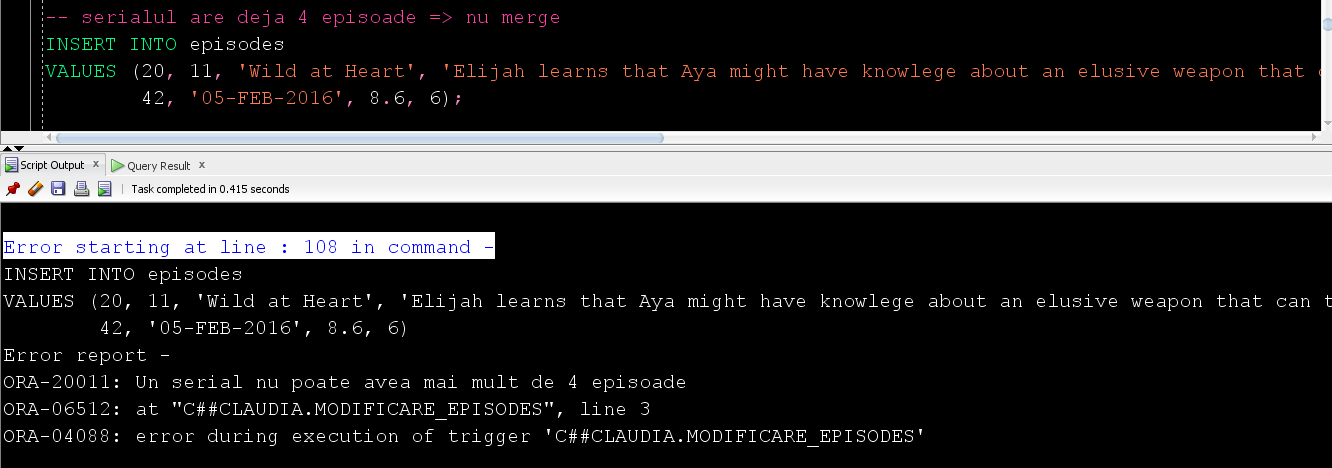
42, '15-OCT-2014', 8.5, 8);

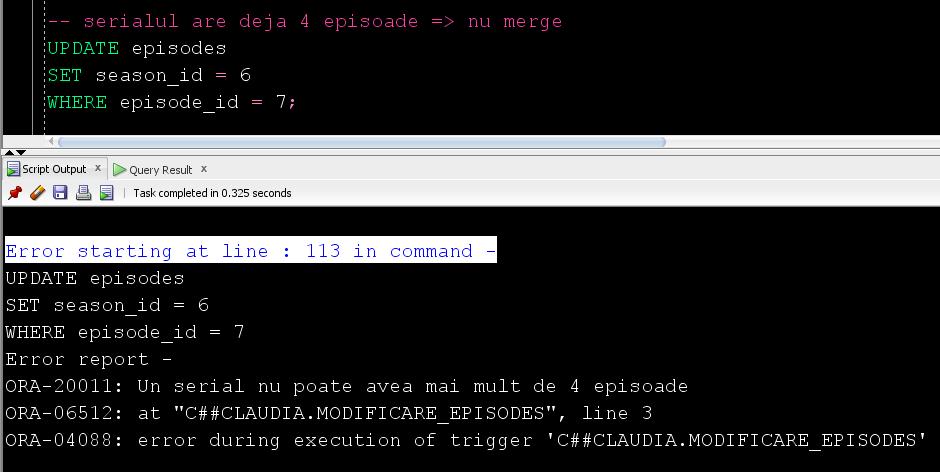
END LOOP;

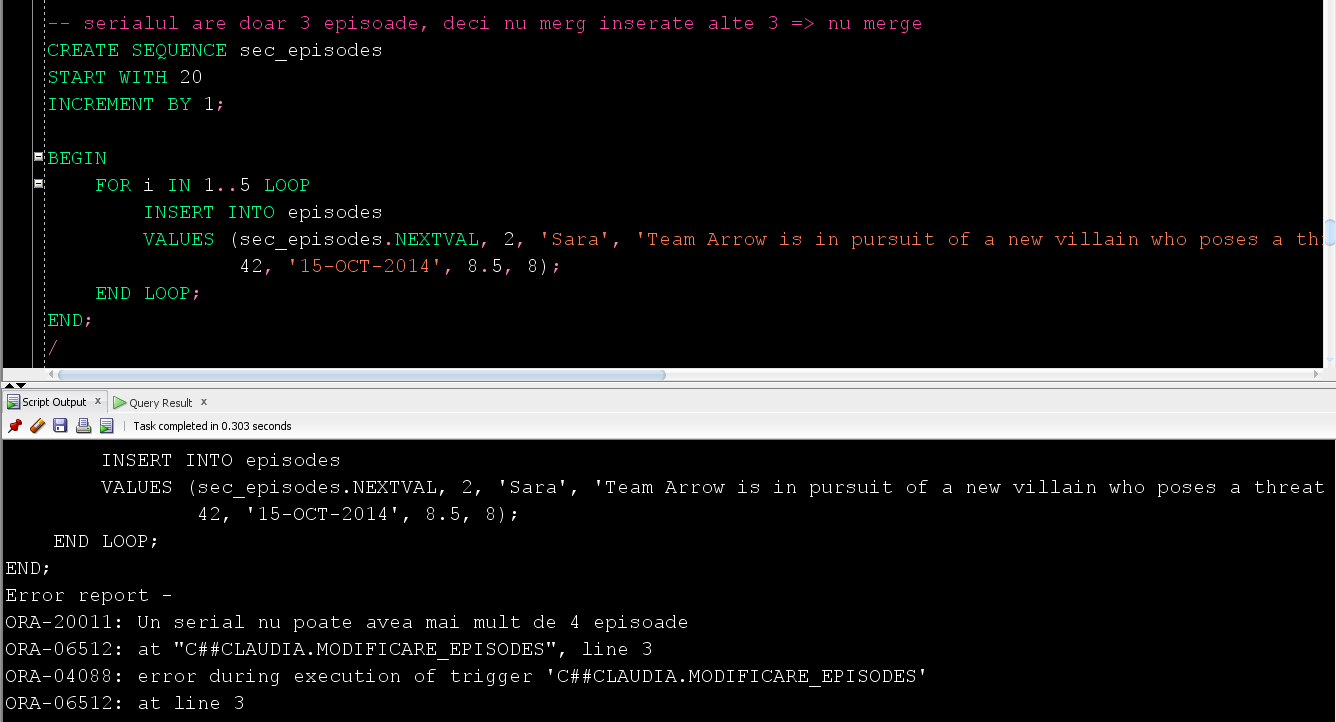
END;

/









# Definirea unui trigger LDD

Pentru fiecare comanda LDD efectuata sa se insereze in tabela istoric\_comenzi numele comenzii, obiectul asupra căreia a fost efectuată, data efectuării și utilizatorul ce a efectuat comanda.

-- creare tabel istoric\_comenzi

CREATE TABLE istoric\_comenzi

(id NUMBER(3) PRIMARY KEY,

comanda VARCHAR2(20),

obiect VARCHAR2(30),

utilizator VARCHAR2(30),

data TIMESTAMP);

-- creare secventa

CREATE SEQUENCE sec\_istoric\_comenzi

START WITH 1

INCREMENT BY 1;

-- creare trigger

CREATE OR REPLACE TRIGGER comenzi\_ldd

AFTER CREATE OR ALTER OR DROP ON SCHEMA

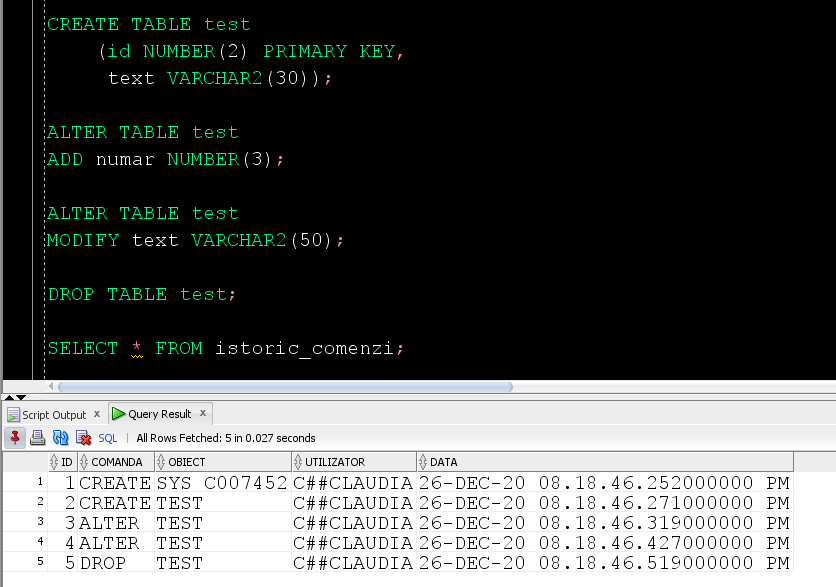
BEGIN

INSERT INTO istoric\_comenzi

VALUES (sec\_istoric\_comenzi.NEXTVAL, SYS.SYSEVENT, SYS.DICTIONARY\_OBJ\_NAME, SYS.LOGIN\_USER, SYSTIMESTAMP);

END;

/



# Definirea unui pachet care sa conțină toate obiectele definite în cadrul proiectului

CREATE OR REPLACE PACKAGE pachet\_1 IS

PROCEDURE modificare\_categorii

(serial series.title%TYPE,

categ1 VARCHAR2,

optiune VARCHAR2,

categ2 VARCHAR2 := NULL);

PROCEDURE afisare\_episoade

(serial series.title%TYPE);

FUNCTION nr\_episoade

(prenume actors.first\_name%TYPE := NULL,

nume actors.last\_name%TYPE := NULL,

inceput DATE,

sfarsit DATE)

RETURN NUMBER;

PROCEDURE afisare\_seriale\_producator

(prenume producers.first\_name%TYPE DEFAULT NULL,

nume producers.last\_name%TYPE DEFAULT NULL);

END pachet\_1;

/

CREATE OR REPLACE PACKAGE BODY pachet\_1 IS

PROCEDURE modificare\_categorii

(serial series.title%TYPE,

categ1 VARCHAR2,

optiune VARCHAR2,

categ2 VARCHAR2 := NULL)

AS

categorii genres;

i INTEGER;

BEGIN

-- obtinere lista categorii pentru seraialul dat

SELECT genre INTO categorii

FROM series

WHERE title = INITCAP(serial);

IF UPPER(optiune) = 'INSERTING' THEN

IF categ1 IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Categoria nou introdusa nu poate sa fie NULL');

ELSE

-- adaugarea unei categorii noi

categorii.extend();

categorii(categorii.last) := INITCAP(categ1);

END IF;

ELSIF UPPER(optiune) = 'DELETING' THEN

IF categ1 IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Categoria de sters nu poate sa fie NULL');

ELSE

-- determinarea pozitiei categoriei ce trebuie stearsa

i := categorii.FIRST;

WHILE i <= categorii.LAST AND categorii(i) <> INITCAP(categ1) LOOP

i := categorii.NEXT(i);

END LOOP;

IF i IS NOT NULL THEN

-- stergerea categoriei

categorii.DELETE(i);

ELSE

RAISE\_APPLICATION\_ERROR(-20002, 'Nu exista categoria introdusa');

END IF;

END IF;

ELSIF UPPER(optiune) = 'UPDATING' THEN

IF categ1 IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Categoria de actualizat nu poate sa fie NULL');

ELSE

IF categ2 IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Categoria nou introdusa nu poate sa fie NULL');

ELSE

-- determinarea pozitiei categoriei ce trebuie modificata

i := categorii.FIRST;

WHILE i <= categorii.LAST AND categorii(i) <> INITCAP(categ1) LOOP

i := categorii.NEXT(i);

END LOOP;

IF i IS NOT NULL THEN

-- odificarea categoriei

categorii(i) := INITCAP(categ2);

ELSE

RAISE\_APPLICATION\_ERROR(-20002, 'Nu exista categoria introdusa');

END IF;

END IF;

END IF;

ELSE

RAISE\_APPLICATION\_ERROR(-20003, 'Optiunea introdusa este gresita');

END IF;

-- actualizare lista categorii

UPDATE series

SET genre = categorii

WHERE title = serial;

DBMS\_OUTPUT.PUT\_LINE('Lista de categorii a fost actualizata cu succes');

EXCEPTION

WHEN no\_data\_found THEN

RAISE\_APPLICATION\_ERROR(-20004, 'Nu exista serial cu numele dat');

WHEN too\_many\_rows THEN

RAISE\_APPLICATION\_ERROR(-20005, 'Exista mai multe seriale cu acest nume');

END;

PROCEDURE afisare\_episoade

(serial series.title%TYPE)

AS

TYPE ref\_cursor IS REF CURSOR;

CURSOR sezoane (id\_serial NUMBER) IS

SELECT season\_number, starting\_date, ending\_date,

CURSOR (SELECT episode\_number, title, description, rating

FROM episodes e

WHERE e.season\_id = s.season\_id)

FROM seasons s

WHERE series\_id = id\_serial;

episoade ref\_cursor;

id\_serial series.series\_id%TYPE;

numar\_sez seasons.season\_number%TYPE;

inceput\_sez seasons.starting\_date%TYPE;

sfarsit\_sez seasons.ending\_date%TYPE;

TYPE ep IS RECORD (numar episodes.episode\_number%TYPE,

titlu episodes.title%TYPE,

descriere episodes.description%TYPE,

rating episodes.rating%TYPE);

episod ep;

exista\_sezoane BOOLEAN := FALSE;

exista\_episoade BOOLEAN;

BEGIN

-- determinare id serial

SELECT series\_id INTO id\_serial

FROM series

WHERE title = serial;

OPEN sezoane(id\_serial);

LOOP

FETCH sezoane INTO numar\_sez, inceput\_sez, sfarsit\_sez, episoade;

EXIT WHEN sezoane%NOTFOUND;

exista\_sezoane := TRUE;

-- afisare sezon

DBMS\_OUTPUT.PUT\_LINE('------------------------------------------');

DBMS\_OUTPUT.PUT('SEZONUL ' || numar\_sez || ': ' || inceput\_sez || ' - ');

IF sfarsit\_sez IS NULL THEN

-- sezonul se afla in derulare

DBMS\_OUTPUT.PUT\_LINE('prezent');

ELSE

DBMS\_OUTPUT.PUT\_LINE(sfarsit\_sez);

END IF;

-- afisare episoade

exista\_episoade := FALSE;

LOOP

FETCH episoade INTO episod;

EXIT WHEN episoade%NOTFOUND;

exista\_episoade := TRUE;

DBMS\_OUTPUT.PUT\_LINE(episod.numar || '. ' || episod.titlu || ' - ' || episod.rating);

DBMS\_OUTPUT.PUT\_LINE('Synopsis: ' || episod.descriere);

DBMS\_OUTPUT.PUT\_LINE('');

END LOOP;

IF NOT exista\_episoade THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista episoade pentru acest sezon');

END IF;

DBMS\_OUTPUT.PUT\_LINE('------------------------------------------');

DBMS\_OUTPUT.PUT\_LINE('');

END LOOP;

CLOSE sezoane;

IF NOT exista\_sezoane THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista sezoane pentru acest serial');

END IF;

EXCEPTION

WHEN no\_data\_found THEN

RAISE\_APPLICATION\_ERROR(-20004, 'Nu exista serial cu numele dat');

WHEN too\_many\_rows THEN

RAISE\_APPLICATION\_ERROR(-20005, 'Exista mai multe seriale cu acest nume');

END;

FUNCTION nr\_episoade

(prenume actors.first\_name%TYPE := NULL,

nume actors.last\_name%TYPE := NULL,

inceput DATE,

sfarsit DATE)

RETURN NUMBER IS

nr\_ep NUMBER(3);

id\_actor actors.actor\_id%TYPE;

BEGIN

IF inceput > sfarsit THEN

RAISE\_APPLICATION\_ERROR(-20006, 'Data de inceput trebuie sa fie mai mica decat data de sfarsit');

RETURN -1;

END IF;

IF prenume IS NULL AND nume IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20007, 'Nu poate sa fie si numele si prenumele NULL');

RETURN -1;

END IF;

-- determinarea id-ului actorului dat

-- (acest pas se face separat ca sa se poata arunca exceptie in cazul in care

-- nu exista actorulul sau exista mai multi acotri cu acest nume)

IF nume IS NOT NULL AND prenume IS NOT NULL THEN

-- numele si prenumele nusunt NULL

SELECT actor\_id INTO id\_actor

FROM actors

WHERE first\_name = prenume

AND last\_name = nume;

ELSIF nume IS NULL AND prenume IS NOT NULL THEN

-- prenumele nu este NULL

SELECT actor\_id INTO id\_actor

FROM actors

WHERE first\_name = prenume;

ELSE

-- numele nu este NULL

SELECT actor\_id INTO id\_actor

FROM actors

WHERE last\_name = nume;

END IF;

SELECT COUNT(\*) INTO nr\_ep

FROM playing p JOIN characters ch ON (p.character\_id = ch.character\_id)

JOIN appearing\_in ap ON (ch.character\_id = ap.character\_id)

JOIN episodes e ON (ap.episode\_id = e.episode\_id)

WHERE p.actor\_id = id\_actor

AND (p.starting\_date <= sfarsit AND p.ending\_date >= inceput)

AND e.airing\_date BETWEEN inceput AND sfarsit;

RETURN nr\_ep;

EXCEPTION

WHEN no\_data\_found THEN

RAISE\_APPLICATION\_ERROR(-20004, 'Nu exista actor cu numele dat');

RETURN -1;

WHEN too\_many\_rows THEN

RAISE\_APPLICATION\_ERROR(-20005, 'Exista mai multi actori cu acest nume');

RETURN -1;

END;

PROCEDURE afisare\_seriale\_producator

(prenume producers.first\_name%TYPE DEFAULT NULL,

nume producers.last\_name%TYPE DEFAULT NULL)

AS

TYPE pers IS RECORD (prenume producers.first\_name%TYPE,

nume producers.last\_name%TYPE);

TYPE tab\_pers IS TABLE OF pers;

CURSOR actori (id\_serial NUMBER) IS

SELECT a.actor\_id, a.first\_name, a.last\_name

FROM actors a JOIN playing p ON(a.actor\_id = p.actor\_id)

WHERE series\_id = id\_serial;

personaje tab\_pers;

id\_producator producers.producer\_id%TYPE;

i INTEGER;

exista\_actori BOOLEAN;

BEGIN

IF prenume IS NULL AND nume IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20007, 'Nu poate sa fie si numele si prenumele NULL');

END IF;

-- determinarea id-ului producatorului dat

-- (acest pas se face separat ca sa se poata arunca exceptie in cazul in care

-- nu exista producatorul sau exista mai multi producatori cu acest nume)

IF nume IS NOT NULL AND prenume IS NOT NULL THEN

-- numele si prenumele nusunt NULL

SELECT producer\_id INTO id\_producator

FROM producers

WHERE first\_name = prenume

AND last\_name = nume;

ELSIF nume IS NULL AND prenume IS NOT NULL THEN

-- prenumele nu este NULL

SELECT producer\_id INTO id\_producator

FROM producers

WHERE first\_name = prenume;

ELSE

-- numele nu este NULL

SELECT producer\_id INTO id\_producator

FROM producers

WHERE last\_name = nume;

END IF;

FOR serial IN (SELECT series\_id, title

FROM produced\_by JOIN series USING (series\_id)

WHERE producer\_id = id\_producator)

LOOP

-- afisare serial

DBMS\_OUTPUT.PUT\_LINE('--- ' || UPPER(serial.title) || ' ---');

-- afisare actori

DBMS\_OUTPUT.PUT\_LINE('--- Actori: ');

exista\_actori := FALSE;

FOR actor in actori(serial.series\_id) LOOP

exista\_actori := TRUE;

DBMS\_OUTPUT.PUT(actor.first\_name || ' ' || actor.last\_name || ' - ');

-- afisare personaje jucate de actor

SELECT c.first\_name, c.last\_name BULK COLLECT INTO personaje

FROM characters c JOIN playing p USING(character\_id)

WHERE actor\_id = actor.actor\_id;

i := personaje.FIRST;

WHILE i<= personaje.LAST LOOP

DBMS\_OUTPUT.PUT(personaje(i).prenume || ' ' || personaje(i).nume);

IF i <> personaje.LAST THEN

DBMS\_OUTPUT.PUT(', ');

END IF;

i := personaje.NEXT(i);

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('');

END LOOP;

IF NOT exista\_actori THEN

-- nu exista actori

DBMS\_OUTPUT.PUT\_LINE('Nu exista actori');

END IF;

DBMS\_OUTPUT.PUT\_LINE('');

DBMS\_OUTPUT.PUT\_LINE('----------------------------------------------------------');

END LOOP;

EXCEPTION

WHEN no\_data\_found THEN

RAISE\_APPLICATION\_ERROR(-20004, 'Nu exista producator cu numele dat');

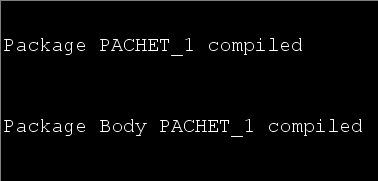
WHEN too\_many\_rows THEN

RAISE\_APPLICATION\_ERROR(-20005, 'Exista mai multi producatori cu acest nume');

END;

END pachet\_1;

/



# Definirea unui pachet care să includă tipuri de date complexe și obiecte necesare pentru acțiuni integrate.

Sa se afișeze informații despre actorii principali ai unui serial (actori care joaca pe durata a cel puțin 75% din durata întregului serial): nume, prenume, data nașterii, numele personajului pe care îl joaca, lista episoadelor în care apare.

CREATE OR REPLACE PACKAGE inf\_actori IS

-- tipuri de date

TYPE rec\_pers IS RECORD (prenume characters.first\_name%TYPE,

nume characters.last\_name%TYPE);

TYPE tab\_pers IS TABLE OF rec\_pers;

TYPE eps IS TABLE OF episodes.title%TYPE;

TYPE rec\_actori IS RECORD (prenume actors.first\_name%TYPE,

nume actors.last\_name%TYPE,

data\_nastere actors.birth\_date%TYPE,

personaje tab\_pers,

episoade eps);

TYPE tab\_actori IS TABLE OF rec\_actori;

-- Obtinere durata in care un actor joaca intr-un serial

FUNCTION screen\_time\_serial

(prenume actors.first\_name%TYPE,

nume actors.last\_name%TYPE,

nume\_serial series.title%TYPE)

RETURN NUMBER;

-- Obtinere despre actorii principali ai unui serial

FUNCTION actori\_principali

(nume\_serial series.title%TYPE,

actori OUT tab\_actori)

RETURN NUMBER;

-- Afisare actorii principali ai unui serial

PROCEDURE afis\_actori\_principali

(nume\_serial series.title%TYPE);

END inf\_actori;

/

CREATE OR REPLACE PACKAGE BODY inf\_actori IS

-- PRIVATE

-- Functie ce determina durata totala a unui serial

FUNCTION durata\_serial

(nume\_serial series.title%TYPE)

RETURN NUMBER IS

durata NUMBER(5);

BEGIN

SELECT SUM(minutes) INTO durata

FROM episodes JOIN seasons USING (season\_id)

JOIN series s USING (series\_id)

WHERE s.title = INITCAP(nume\_serial);

RETURN durata;

END;

-- Functie ce determina durata in care un actor joaca intr-un sezon

FUNCTION screen\_time\_sez

(prenume actors.first\_name%TYPE,

nume actors.last\_name%TYPE,

nume\_serial series.title%TYPE,

nr\_sez seasons.season\_number%TYPE)

RETURN NUMBER IS

screen\_time NUMBER(5);

BEGIN

SELECT SUM(minutes) INTO screen\_time

FROM episodes JOIN seasons USING(season\_id)

JOIN series s USING(series\_id)

JOIN appearing\_in USING (episode\_id)

JOIN characters USING (character\_id)

JOIN playing USING (character\_id)

JOIN actors a USING (actor\_id)

WHERE s.title = INITCAP(nume\_serial)

AND season\_number = nr\_sez

AND a.first\_name = INITCAP(prenume)

AND a.last\_name = INITCAP(nume);

RETURN NVL(screen\_time, 0);

END;

-- PUBLIC

FUNCTION screen\_time\_serial

(prenume actors.first\_name%TYPE,

nume actors.last\_name%TYPE,

nume\_serial series.title%TYPE)

RETURN NUMBER IS

screen\_time NUMBER(5) := 0;

id\_actor actors.actor\_id%TYPE;

id\_serial series.series\_id%TYPE;

identif NUMBER := 1;

BEGIN

-- pentru a verifica daca actorul exista

SELECT actor\_id INTO id\_actor

FROM actors

WHERE first\_name = prenume

AND last\_name = nume;

identif := 2;

-- pentru a verifica daca serialul exista

SELECT series\_id INTO id\_serial

FROM series

WHERE title = INITCAP(nume\_serial);

-- determinare screen\_time\_sez pt actorul pt fiecare sezon al serialului

FOR sez IN (SELECT season\_number

FROM seasons JOIN series s USING (series\_id)

WHERE s.title = INITCAP(nume\_serial))

LOOP

screen\_time := screen\_time + screen\_time\_sez(prenume, nume, nume\_serial, sez.season\_number);

END LOOP;

RETURN screen\_time;

EXCEPTION

WHEN no\_data\_found THEN

IF identif = 1 THEN

RAISE\_APPLICATION\_ERROR(-20009, 'Numele actorului dat nu este bun');

ELSE

RAISE\_APPLICATION\_ERROR(-20010, 'Numele serialului dat nu este bun');

END IF;

RETURN -1;

END;

FUNCTION actori\_principali

(nume\_serial series.title%TYPE,

actori OUT tab\_actori)

RETURN NUMBER IS

TYPE rec\_act IS RECORD (prenume actors.first\_name%TYPE,

nume actors.last\_name%TYPE,

data\_nastere actors.birth\_date%TYPE);

TYPE tab\_act IS TABLE OF rec\_act;

v\_actori tab\_act;

v\_pers tab\_pers;

v\_eps eps;

nr\_act NUMBER(2) := 0;

id\_serial series.series\_id%TYPE;

BEGIN

-- pentru a verifica daca serialul exista

SELECT series\_id INTO id\_serial

FROM series

WHERE title = INITCAP(nume\_serial);

actori := tab\_actori();

-- obtinere actorii ce joaca in serialul dat

SELECT a.first\_name, a.last\_name, a.birth\_date BULK COLLECT INTO v\_actori

FROM actors a JOIN playing USING (actor\_id)

JOIN series USING (series\_id)

WHERE title = INITCAP(nume\_serial);

FOR i IN v\_actori.FIRST..v\_actori.LAST LOOP

IF screen\_time\_serial(v\_actori(i).prenume, v\_actori(i).nume, nume\_serial) >= 0.75 \* durata\_serial(nume\_serial) THEN

nr\_act := nr\_act + 1;

-- obtinere personajele jucate de actor

SELECT first\_name, last\_name BULK COLLECT INTO v\_pers

FROM characters JOIN playing USING(character\_id)

WHERE actor\_id = (SELECT actor\_id

FROM actors

WHERE first\_name = v\_actori(i).prenume

AND last\_name = v\_actori(i).nume)

AND series\_id = (SELECT series\_id

FROM series

WHERE title = INITCAP(nume\_serial));

-- obtinere lista episoade in care joaca actorul

SELECT e.title BULK COLLECT INTO v\_eps

FROM episodes e JOIN seasons USING(season\_id)

JOIN series s USING(series\_id)

JOIN appearing\_in USING (episode\_id)

JOIN characters USING (character\_id)

JOIN playing USING (character\_id)

JOIN actors a USING (actor\_id)

WHERE s.title = INITCAP(nume\_serial)

AND a.first\_name = v\_actori(i).prenume

AND a.last\_name = v\_actori(i).nume;

actori.EXTEND;

actori(actori.LAST).prenume := v\_actori(i).prenume;

actori(actori.LAST).nume := v\_actori(i).nume;

actori(actori.LAST).data\_nastere := v\_actori(i).data\_nastere;

actori(actori.LAST).personaje := v\_pers;

actori(actori.LAST).episoade := v\_eps;

END IF;

END LOOP;

RETURN nr\_act;

EXCEPTION

WHEN no\_data\_found THEN

RAISE\_APPLICATION\_ERROR(-20010, 'Numele serialului dat nu este bun');

RETURN -1;

END;

PROCEDURE afis\_actori\_principali

(nume\_serial series.title%TYPE)

IS

actori tab\_actori;

nr\_act NUMBER(2);

BEGIN

nr\_act := actori\_principali(nume\_serial, actori);

DBMS\_OUTPUT.PUT\_LINE('Serialul ' || INITCAP(nume\_serial) || ' are ' || nr\_act || ' actori principali');

FOR i IN 1..nr\_act LOOP

DBMS\_OUTPUT.PUT\_LINE(i || '. ' || actori(i). prenume || ' ' || actori(i).nume || ' - ' || actori(i).data\_nastere);

DBMS\_OUTPUT.PUT\_LINE('-- Personaje jucate: ');

FOR j IN actori(i).personaje.FIRST..actori(i).personaje.LAST LOOP

DBMS\_OUTPUT.PUT\_LINE(actori(i).personaje(j).prenume || ' ' || actori(i).personaje(j).nume);

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('-- Episoade in care apare: ');

FOR j IN actori(i).episoade.FIRST..actori(i).episoade.LAST LOOP

DBMS\_OUTPUT.PUT\_LINE(actori(i).episoade(j));

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('');

END LOOP;

END;

END inf\_actori;

/

