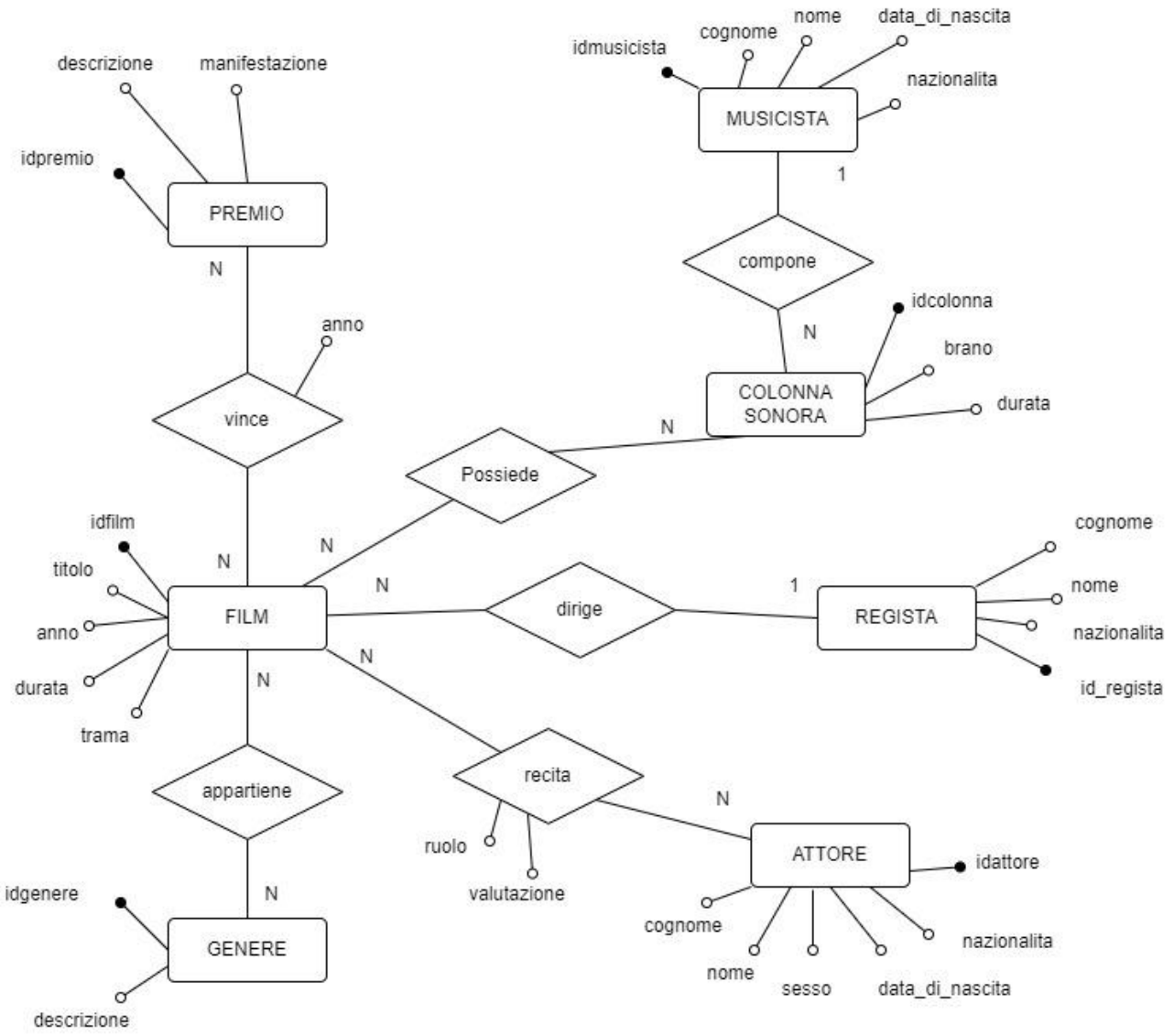
SOLUZIONE TEST FINALE

ESERCIZIO 1

SOLUZIONE SCHEMA E/R BANCA DATI CINEMATOGRAFICA



NOTA:

Volendo era possibile inserire l'entità NAZIONE collegata 1:N alle entita: REGISTA, ATTORE, MUSICISTA

SCHEMA RELAZIONALE

1) **generi** (idgenere (PK), descrizione)
2) **musicisti** (idmusicista (PK), cognome, nome, data_di_nascita, nazionalita)
3) **premi** (idpremio (PK), descrizione,manifestazione)
4) **attori** (idattore (PK), cognome, nome, nazionalita, sesso, data_di_nascita)
5) **registi** (idregista (PK), nome, cognome, nazionalita)
6) **colonnesonore** (idcolonna, brano, durata, idmusicista (FK))
7) **film** (idfilm (PK), titolo, anno, durata, trama, idregista (FK))
8) **film_attori** (id (PK), id_film (FK), id_attore, ruolo, valutazione)
9) **film_colonnesonore** (id (PK), idfilm (FK), idcolonna (FK))
10) **film_generi** (id (PK), idfilm (FK), idgenere (FK))
11) **film_premi** (id (PK), idfilm (FK), idpremio (FK), anno)

SCHEMA FISICO

```
CREATE SCHEMA IF NOT EXISTS FILM;
USE FILM;
CREATE TABLE IF NOT EXISTS generi (
    idgenere INT AUTO_INCREMENT  PRIMARY KEY,
    descrizione  VARCHAR(30) NOT NULL
);

CREATE TABLE IF NOT EXISTS musicisti(
    idmusicista INT AUTO_INCREMENT  PRIMARY KEY,
    cognome  VARCHAR(30) NOT NULL,
    nome  VARCHAR(30) NOT NULL,
    data_nascita DATE NOT NULL,
    nazionalita VARCHAR(30)
);

CREATE TABLE IF NOT EXISTS premi(
    idpremio INT AUTO_INCREMENT  PRIMARY KEY,
    descrizione  VARCHAR(50) NOT NULL,
    manifestazione  VARCHAR(50) NOT NULL
);

CREATE TABLE IF NOT EXISTS attori(
    idattore INT AUTO_INCREMENT  PRIMARY KEY,
    cognome  VARCHAR(30) NOT NULL,
    nome  VARCHAR(30) NOT NULL,
    data_nascita DATE NOT NULL,
    nazionalita VARCHAR(30)
);
```

```
CREATE TABLE IF NOT EXISTS colonnesonore(
    idcolonnasonora INT AUTO_INCREMENT  PRIMARY KEY,
    brano VARCHAR(30) NOT NULL,
    durata  SMALLINT NOT NULL,
    idmusicista INT NOT NULL,
    FOREIGN KEY (idmusicista) REFERENCES musicisti(idmusicista)
);

CREATE TABLE IF NOT EXISTS film(
    idfilm INT AUTO_INCREMENT  PRIMARY KEY,
    titolo VARCHAR(80) NOT NULL,
    anno  YEAR NOT NULL,
    durata  SMALLINT NOT NULL,
    trama VARCHAR(255) NOT NULL,
    idregista INT NOT NULL,
    FOREIGN KEY (idregista) REFERENCES registi(idregista)
);

CREATE TABLE IF NOT EXISTS film_attori(
    id INT AUTO_INCREMENT  PRIMARY KEY,
    idfilm INT NOT NULL,
    idattore  INT NOT NULL,
    ruolo VARCHAR(30) NOT NULL,
    valutazione TINYINT NOT NULL,
    FOREIGN KEY (idfilm) REFERENCES film(idfilm),
    FOREIGN KEY (idattore) REFERENCES attori(idattore)
);

CREATE TABLE IF NOT EXISTS film_colonnesonore(
    id INT AUTO_INCREMENT  PRIMARY KEY,
    idfilm INT NOT NULL,
    idcolonnasonora  INT NOT NULL,
    FOREIGN KEY (idfilm) REFERENCES film(idfilm),
    FOREIGN KEY (idcolonnasonora) REFERENCES colonnesonore(idcolonnasonora)
);

CREATE TABLE IF NOT EXISTS film_generi(
    id INT AUTO_INCREMENT  PRIMARY KEY,
    idfilm INT NOT NULL,
    idgenere  INT NOT NULL,
    FOREIGN KEY (idfilm) REFERENCES film(idfilm),
    FOREIGN KEY (idgenere) REFERENCES attori(idgenere)
);
```

```
CREATE TABLE IF NOT EXISTS film_premi(
    id INT AUTO_INCREMENT  PRIMARY KEY,
    idfilm INT NOT NULL,
    idpremio  INT NOT NULL,
    anno   YEAR NOT NULL,
    FOREIGN KEY (idfilm) REFERENCES film(idfilm),
    FOREIGN KEY (idpremio) REFERENCES premi(idpremio)
);
```

ESERCIZIO 2

1) INSERT INTO utenti (cognome, nome, cellulare, email) VALUES (‘Rossi’, ‘Giulio’, ‘336574632’, ‘giuliorossi@gmail.com’);
2) UPDATE libri SET anno = 2020 WHERE id_libro = 10)
3) UPDATE prestiti SET data_restituzione = ‘2022-12-21’ WHERE id_prestito = 200)

4) SELECT COUNT(*) AS num_prestiti FROM prestiti WHERE data_inizio BETWEEN ‘2022-11-01’ AND ‘2022-11-30’
5) SELECT L.* FROM libri L INNER JOIN scaffafi S ON L.id_libro = S.id_libro WHERE stanza = 7 AND armadio = 2ORDER BY L.titolo
6) SELECT U.* FROM utenti INNER JOIN prestiti P ON U.id_utente = P.id_utente WHERE data_inizio = ‘2022-12-20’ ORDER BY U.cognome, U.nome
7) SELECT genere, COUNT(*) AS num_libri FROM libri GROUP BY genere ORDER BY genere

8) SELECT titolo, genere FROM libri WHERE anno = (SELECT MAX(anno) FROM libri)
9) SELECT U.*, COUNT(*) AS num_prestiti
   FROM utenti INNER JOIN prestiti P ON U.id_utente = P.id_utente
   GROUP BY U.id_utente
   HAVING COUNT(*) >= ALL (SELECT COUNT(*) FROM prestiti GROUP BY id_utente)