**База знаний о сериалах и кино.** Волков Александр M3437

### Создание

```
------------
-- Types --
------------

CREATE TYPE Currency as enum (
    'USD',
    'RUB',
    'EUR'
);

CREATE TYPE MoneyType as (
    Qnt BIGINT,
    Currency Currency
);

CREATE TYPE FilmType as enum (
    'film',
    'series'
);

------------
-- Tables --
------------

CREATE TABLE Country(
    CountryId INT NOT NULL,
    CountryName VARCHAR(100) NOT NULL,
    PRIMARY KEY(CountryId)
);
CREATE INDEX ON Country USING (CountryName, CountryId);

CREATE TABLE Person(
    PersonId INT NOT NULL,
    FirstName VARCHAR(100) NOT NULL,
    Birthday DATE NOT NULL,
    LastName VARCHAR(100),
    Patronymic VARCHAR(100),
    CountryId INT,
    PRIMARY KEY (PersonId),
    FOREIGN KEY (CountryId) REFERENCES Country(CountryId)
);
CREATE INDEX ON Person USING HASH (CountryId);

CREATE TABLE Genre(
    GenreId INT NOT NULL,
    GenreName VARCHAR(100) NOT NULL,
    PRIMARY KEY (GenreId)
);

CREATE TABLE Film(
    FilmId INT NOT NULL,
    FilmName VARCHAR(100) NOT NULL,
    ReleaseDate DATE NOT NULL,
    Duration INTERVAL NOT NULL,
    FType FilmType NOT NULL,
```

```
    Slogan VARCHAR(200),
    Budget MoneyType,
    Description VARCHAR(1000),
    PRIMARY KEY (FilmId)
);
CREATE INDEX ON Film USING BTREE(ReleaseDate);

CREATE TABLE FilmGenre(
    FilmId INT NOT NULL,
    GenreId INT NOT NULL,
    PRIMARY KEY (FilmId, GenreId),
    FOREIGN KEY (FilmId) REFERENCES Film(FilmId),
    FOREIGN KEY (GenreId) REFERENCES Genre(GenreId)
);
CREATE INDEX ON FilmGenre USING HASH (FilmId);
CREATE INDEX ON FilmGenre USING HASH (GenreId);

CREATE TABLE FilmCountry(
    FilmId INT NOT NULL,
    CountryId INT NOT NULL,
    PRIMARY KEY (FilmId, CountryId),
    FOREIGN KEY (FilmId) REFERENCES Film(FilmId),
    FOREIGN KEY (CountryId) REFERENCES Country(CountryId)
);
CREATE INDEX ON FilmCountry USING HASH (FilmId);
CREATE INDEX ON FilmCountry USING HASH (CountryId);

CREATE TABLE Score(
    FilmId INT NOT NULL,
    UserId INT NOT NULL,
    Score FLOAT NOT NULL,
    PRIMARY KEY (FilmId, UserId),
    FOREIGN KEY (FilmId) REFERENCES Film(FilmId),
    CHECK (Score BETWEEN 0 AND 10)
);
CREATE INDEX ON Score USING HASH (FilmId);

CREATE TABLE Series(
    FilmId INT NOT NULL,
    Season INT NOT NULL,
    SeriesNo INT NOT NULL,
    SeriesName VARCHAR(100) NOT NULL,
    ReleaseDate DATE NOT NULL,
    PRIMARY KEY (FilmId, Season, SeriesNo),
    FOREIGN KEY (FilmId) REFERENCES Film(FilmId)
);
CREATE INDEX ON Series USING HASH (FilmId);
CREATE INDEX ON Series USING BTREE (ReleaseDate);

CREATE TABLE Boxoffice(
    FilmId INT NOT NULL,
    CountryId INT NOT NULL,
    Income MoneyType NOT NULL,
    PRIMARY KEY (FilmId, CountryId),
    FOREIGN KEY (FilmId) REFERENCES Film(FilmId),
    FOREIGN KEY (CountryId) REFERENCES Country(CountryId)
);
CREATE INDEX ON Boxoffice USING HASH (FilmId);
CREATE INDEX ON Boxoffice USING HASH (CountryId);
```

```
CREATE TABLE Profession(
    ProfessionId INT NOT NULL,
    ProfessionName VARCHAR(100) NOT NULL,
    PRIMARY KEY (ProfessionId)
);

CREATE TABLE Filmmaker(
    FilmId INT NOT NULL,
    PersonId INT NOT NULL,
    ProfessionId INT NOT NULL,
    Character VARCHAR(100),
    PRIMARY KEY (FilmId, PersonId, ProfessionId),
    FOREIGN KEY (FilmId) REFERENCES Film(FilmId),
    FOREIGN KEY (PersonId) REFERENCES Person(PersonId),
    FOREIGN KEY (ProfessionId) REFERENCES Profession(ProfessionId)
);
CREATE INDEX ON Filmmaker USING HASH (FilmId);
CREATE INDEX ON Filmmaker USING HASH (PersonId);
CREATE INDEX ON Filmmaker USING HASH (ProfessionId);
CREATE INDEX ON Filmmaker(PersonId, FilmId);
CREATE INDEX ON Filmmaker(FilmId, PersonId);

CREATE TABLE AwardingOrg(
    AwardId INT NOT NULL,
    AwardName VARCHAR(100) NOT NULL,
    AwardDate DATE NOT NULL,
    PRIMARY KEY (AwardId)
);
CREATE INDEX ON AwardingOrg USING HASH (AwardName);
CREATE INDEX ON AwardingOrg USING BTREE (AwardDate);

CREATE TABLE FilmNomination(
    FNominationId INT NOT NULL,
    FNominationName VARCHAR(100) NOT NULL,
    PRIMARY KEY (FNominationId)
);

CREATE TABLE PersonNomination(
    PNominationId INT NOT NULL,
    PNominationName VARCHAR(100) NOT NULL,
    ProfessionId INT,
    PRIMARY KEY (PNominationId),
    FOREIGN KEY (ProfessionId) REFERENCES Profession(ProfessionId)
);
CREATE INDEX ON PersonNomination USING HASH (ProfessionId);

CREATE TABLE FilmAward(
    AwardId INT NOT NULL,
    FNominationId INT NOT NULL,
    FilmId INT NOT NULL,
    Win BOOLEAN NOT NULL,
    PRIMARY KEY (AwardId, FNominationId, FilmId),
    FOREIGN KEY (FilmId) REFERENCES Film(FilmId),
    FOREIGN KEY (AwardId) REFERENCES AwardingOrg(AwardId),
    FOREIGN KEY (FNominationId) REFERENCES FilmNomination(FNominationId)
);
CREATE INDEX ON FilmAward USING HASH (FilmId);
CREATE INDEX ON FilmAward USING HASH (AwardId);
```

```
CREATE INDEX ON FilmAward USING HASH (FNominationId);

CREATE TABLE PersonAward(
    AwardId INT NOT NULL,
    PNominationId INT NOT NULL,
    FilmId INT NOT NULL,
    PersonId INT NOT NULL,
    ProfessionId INT,
    Win BOOLEAN NOT NULL,
    PRIMARY KEY (AwardId, PNominationId, FilmId, PersonId, ProfessionId),
    FOREIGN KEY (FilmId, PersonId, ProfessionId) REFERENCES Filmmaker(FilmId,
PersonId, ProfessionId),
    FOREIGN KEY (AwardId) REFERENCES AwardingOrg(AwardId),
    FOREIGN KEY (PNominationId) REFERENCES PersonNomination(PNominationId)
);
CREATE INDEX ON PersonAward(FilmId, PersonId, ProfessionId);
CREATE INDEX ON PersonAward USING HASH (AwardId);
CREATE INDEX ON PersonAward USING HASH (PNominationId);

CREATE TABLE Keys(
    TableName VARCHAR(50) NOT NULL,
    MinKey INT NOT NULL,
    PRIMARY KEY (TableName)
);
```

**Заполнение**

```
INSERT INTO Country
    (CountryId, CountryName)
VALUES
    (1, 'United States of America (USA)'),
    (2, 'Russia'),
    (3, 'France'),
    (4, 'United Kingdom (UK)'),
    (5, 'Germany');

INSERT INTO Person
    (PersonId,
    FirstName,
    LastName,
    Birthday,
    Patronymic,
    CountryId)
VALUES
    (1, 'Tom', 'Hanks', '1956-07-09', null, 1),
    (2, 'Robert', 'Zemeckis', '1951-05-14', null, 1),
    (3, 'Ramin', 'Djawadi', '1974-07-19', null, 5),
    (4, 'David', 'Benioff', '1970-09-25', null, 1),
    (5, 'D.B', 'Weiss', '1971-06-23', null, 1),
    (6, 'Peter', 'Dinklage', '1969-06-11', null, 1),
    (7, 'Lena', 'Headey', '1973-10-03', null, 4),
    (8, 'Deborah', 'Riley', '1973-05-16', null, 4),
    (9, 'Эльдар', 'Рязанов', '1927-11-18', 'Александрович', 2),
    (10, 'Вадим', 'Алисов', '1941-02-20', 'Валентинович', 2),
    (11, 'Лазарь', 'Милькис', '1923-11-15', 'Наумович', 2),
    (12, 'Лариса', 'Гузеева', '1959-05-23', 'Андреевна', 2),
    (13, 'Никита', 'Михалков', '1945-10-21', 'Сергеевич', 2),
    (14, 'Алиса', 'Фрейндлих', '1934-12-08', 'Брунова', 2);

INSERT INTO Genre
```

```
    (GenreId, GenreName)
VALUES
    (1, 'фэнтези'),
    (2, 'драма'),
    (3, 'мелодрама'),
    (4, 'боевик'),
    (5, 'военное'),
    (6, 'детектив');

INSERT INTO Film
    (FilmId,
    FilmName,
    ReleaseDate,
    Duration,
    FType,
    Slogan,
    Budget,
    Description)
VALUES
     (1,
    'Forrest Gump',
    '1994-06-23',
    '142m',
    'film',
    'Мир уже никогда не будет прежним, после того как вы увидите его глазами
Форреста Гампа',
     (55000000, 'USD'),
    'От лица главного героя Форреста Гампа, слабоумного безобидного человека
с благородным и открытым сердцем, рассказывается история его необыкновенной
жизни'
     ),
    (2,
    'Game of Thrones',
    '2011-05-21',
    '55m',
    'series',
    'Победа или смерть',
    null,
    'К концу подходит время благоденствия, и лето, длившееся почти
десятилетие, угасает. Вокруг средоточия власти Семи королевств, Железного
трона, зреет заговор'
    ),
    (3,
    'Жестокий романс',
    '1984-10-23',
    '137m',
    'film',
    null,
    null,
    'Действие разворачивается на берегу Волги в вымышленном провинциальном
городке Бряхимове в 1877-1878 годах.'
    ),
    (4,
    'Knives Out',
    '2019-12-12',
    '130m',
    'film',
    'У каждого свой мотив',
    (40000000, 'USD'),
```

```
    null),
    (5,
    'The Lighthouse',
    '2020-05-19',
    '109m',
    'film',
    null,
    (1000000, 'USD'),
    null);

INSERT INTO FilmGenre
    (FilmId, GenreId)
VALUES
    (1, 2),
    (1, 4),
    (1, 5),
    (2, 1),
    (2, 2),
    (3, 3),
    (3, 2),
    (4, 6),
    (5, 2),
    (5, 5);

INSERT INTO FilmCountry
    (FilmId, CountryId)
VALUES
    (1, 1),
    (2, 1),
    (2, 4),
    (3, 2),
    (4, 1);

INSERT INTO Series
    (FilmId,
    Season,
    SeriesNo,
    SeriesName,
    ReleaseDate)
VALUES
    (2, 1, 1, 'Winter Is Coming', '2011-06-14'),
    (2, 1, 2, 'The Kingsroad', '2011-06-24'),
    (2, 1, 3, 'Lord Snow', '2011-07-01'),
    (2, 1, 4, 'Cripples, Bastards, and Broken Things', '2011-07-08'),
    (2, 1, 5, 'The Wolf and the Lion', '2011-07-15'),
    (2, 1, 6, 'A Golden Crown', '2011-07-22'),
    (2, 1, 7, 'You Win or You Die', '2011-07-29'),
    (2, 1, 8, 'The Pointy End', '2011-08-05'),
    (2, 1, 9, 'Baelor', '2011-08-12'),
    (2, 1, 10, 'Fire and Blood', '2011-08-19'),
    (2, 2, 1, 'The North Remembers', '2012-06-01'),
    (2, 3, 1, 'Valar Dohaeris', '2013-05-31'),
    (2, 4, 1, 'Two Swords', '2014-06-06'),
    (2, 5, 1, 'The Wars to Come', '2015-06-12'),
    (2, 6, 1, 'The Red Woman', '2016-06-24'),
    (2, 7, 1, 'Dragonstone', '2017-06-16'),
    (2, 8, 1, 'Winterfell', '2019-06-14');

INSERT INTO Score
```

```
    (FilmId, UserId, Score)
VALUES
    (1, 1, 7),
    (1, 2, 8.5),
    (1, 3, 6.5),
    (2, 1, 6.5),
    (2, 2, 7.8),
    (2, 3, 9.0),
    (3, 1, 8.0),
    (3, 2, 8.7),
    (3, 3, 7.2);

INSERT INTO Boxoffice
    (FilmId, CountryId, Income)
VALUES
    (1, 1, (340000000, 'USD')),
    (1, 5, (5000000, 'EUR')),
    (3, 2, (22000000, 'RUB')),
    (4, 1, (130000000, 'USD'));

INSERT INTO Profession
    (ProfessionId, ProfessionName)
VALUES
    (1, 'режиссер'),
    (2, 'сценарист'),
    (3, 'актер'),
    (4, 'продюсер'),
    (5, 'директор фильма'),
    (6, 'композитор'),
    (7, 'художник по костюмах'),
    (8, 'оператор');

INSERT INTO Filmmaker
    (FilmId, PersonId, ProfessionId, Character)
VALUES
    (1, 1, 3, 'Forrest Gump'),
    (1, 2, 1, null),
    (2, 3, 6, null),
    (2, 4, 4, null),
    (2, 4, 1, null),
    (2, 4, 2, null),
    (2, 5, 4, null),
    (2, 5, 1, null),
    (2, 5, 2, null),
    (2, 6, 3, 'Tyrion Lannister'),
    (2, 7, 3, 'Cersei Lannister'),
    (2, 8, 7, null),
    (3, 9, 1, null),
    (3, 10, 8, null),
    (3, 11, 5, null),
    (3, 12, 3, 'Лариса Дмитриевна Огудалова'),
    (3, 13, 3, 'Сергей Сергеевич Паратов'),
    (3, 14, 3, 'Харита Игнатьевна Огудалова');

INSERT INTO AwardingOrg
    (AwardId, AwardName, AwardDate)
VALUES
    (1, 'Оскар', '1995-05-27'),
    (2, 'Золотой глобус', '1995-01-21'),
```

```
    (3, 'Сатурн', '2019-05-10'),
    (4, 'Премия канала MTV', '1995-03-24'),
    (5, 'Эмми', '2019-04-01');

INSERT INTO FilmNomination
    (FNominationId, FNominationName)
VALUES
    (1, 'ТВ-шоу года'),
    (2, 'Лучший телесериал в жанре фентези'),
    (3, 'Лучший драматический сериал'),
    (4, 'Лучшая драка'),
    (5, 'Лучший фильм'),
    (6, 'Лучший монтаж'),
    (7, 'Лучший адаптивный сценарий'),
    (8, 'Лучшие визуальные эффекты'),
    (9, 'Лучший грим'),
    (10, 'Лучшая работа оператора');

INSERT INTO PersonNomination
    (PNominationId, PNominationName, ProfessionId)
VALUES
    (1, 'Лучшая мужская роль', 3),
    (2, 'Лучший режиссер', 1),
    (3, 'Лучший продюссер', 4),
    (4, 'Лучшая телеакриса', 3),
    (5, 'Лучший актер второго плана в телесериале', 3),
    (6, 'Лучший оператор', 8);

INSERT INTO FilmAward
    (AwardId, FNominationId, FilmId, Win)
VALUES
    (1, 5, 1, True),
    (1, 7, 1, True),
    (1, 9, 1, False),
    (1, 10, 1, False),
    (2, 5, 1, True),
    (2, 8, 1, False),
    (4, 5, 1, False),
    (4, 1, 2, True),
    (4, 4, 2, False),
    (3, 2, 2, True),
    (5, 3, 2, True);

INSERT INTO PersonAward
    (AwardId, PNominationId, FilmId, PersonId, ProfessionId, Win)
VALUES
    (1, 1, 1, 1, 3, True),
    (1, 2, 1, 2, 1, True),
    (2, 1, 1, 1, 3, True),
    (2, 2, 1, 2, 1, False),
    (3, 4, 2, 7, 3, True),
    (3, 5, 2, 6, 3, False),
    (5, 5, 2, 6, 3, True),
    (5, 3, 2, 4, 4, True);

INSERT INTO Keys
    (TableName, MinKey)
VALUES
    ('Country', 6),
```

```
('Person', 20),
('Genre', 20),
('Film', 10),
('Profession', 10),
('AwardingOrg', 10),
('PersonNomination', 10),
('FilmNomination', 11);
```

**Выделим ФЗ:**

1) Простые ФЗ типа *Id -> *Name у таблиц:  Country, Genre, FilmNomination, PersonNomination, Profession, AwardingOrg
2) ФЗ, получившиеся в результате создания таблиц для отношения “многие ко многим” FilmGenre, FilmCountry
3) Person:
   a) PersonId -> FirstName
   b) PersonId -> LastName
   c) PersonId -> Patronymic
   d) PersonId -> Birthday
   e) PersonId -> CountryId

   Ключ отношения – суррогатный PersonId

4) Film:
   a) FilmId -> FilmName
   b) FilmId -> ReleaseDate
   c) FilmId -> Slogan
   d) FilmId -> BudgetQnt
   e) FilmId -> BudgetCurrency
   f) FilmId -> Duration
   g) FilmId -> Description
   h) FilmId -> Type
   i) FilmId -> GenreId

   Ключ отношения – суррогатный FilmId
5) Score:
   a) FilmId, UserId -> Score

   Ключ отношения – FilmId, UserId
6) Series:
   a) FilmId, Season, SeriesNo -> ReleaseDate
   b) FilmId, Season, SeriesNo -> SeriesName

   Ключ отношения - FilmId, Season, SeriesNo
7) Boxoffice:
   a) FilmId, CountryId -> IncomeQnt
   b) FilmId, CountryId -> IncomeCurrency

   Ключ отношения – FilmId, CountryId
8) Filmmaker:
   a) FilmId, PersonId, ProfessionId -> Character

   Ключ отношения – FilmId, PersonId, ProfessionId
9) FilmAward:

a) AwardId, FNominationId, FilmId -> Win

Ключ отношения – AwardId, FNominationId, FilmId

10) PersonNomination:
    a) PNominationId -> ProfessionId

    Ключ отношения - PNominationId
11) PersonAward:
    a) AwardId, PNominationId, FilmId, PersonId, ProfessionId -> Win

    Ключ отношения AwardId, PNominationId, FilmId, PersonId, ProfessionId

**Нормализация**

**1. 1НФ**

Все отношения находятся в 1НФ:

- В них нет повторяющихся групп
- Атрибуты атомарны
- Ключ отношения

**2. 2НФ**

В отношениях отсутствуют атрибута зависящие от части ключа, следовательно все отношения находятся в 2НФ:

**3. 3НФ**

Все отношения в 3НФ т.к. все неключевые атрибуты напрямую зависят от ключей.

**4. НФБК**

Отношения находятся в НФБК, потому что все зависимости уз двух атрибутов уже находятся в НФБК. В остальных отношениях левая часть является надключом.

**5. 4НФ**

Проверим что отношения находятся в 4 НФ. Все отношения из 2 атрибутов уже находятся в 4НФ. Проверим отношения 5, 6, 7, 8, 9, 11. Во всех указанных зависимостях каждый атрибут зависит от ключа и для каждого $X \twoheadrightarrow Y|Z$, X является надключом. Значит все отношения находятся в 4НФ.

**6. 5НФ**

Порверим что отношения уже в 5НФ

Отношения из двух атрибутов $A \rightarrow B$ уже в 5НФ:

- Имеет одну ФЗ которая МЗ
- $A \twoheadrightarrow B \mid \emptyset$ тогда по т. Фейгина *{AB, A} – единственная зависимость соедния при этом каждый её элемнт – надключ.

Во всех остальных отношениях вида

$A_1, A_2, \dots A_n \rightarrow C$: $A_1, A_2, \dots A_n \twoheadrightarrow C \mid \emptyset$

По т. Фейгина $\{A_1\ A_2\ \dots A_n,\ A_1\ A_2\ \dots A_n C\}$ – каждый из элементов этого множества является надключем => все отношения находятся в 5НФ.

**Представления**

```
-----------
-- Views --
-----------

--- Топ 250 самых высокооцененных фильмов
CREATE VIEW TopFilms AS
    SELECT FilmName, FilmScore
    FROM Film F LEFT JOIN
        (SELECT FilmId, AVG(Score) AS FilmScore FROM Score
        GROUP BY FilmId
        ORDER BY FilmScore DESC) AS A
    ON A.FilmId = F.FilmId WHERE F.FType = 'film' LIMIT 250;

--- Самые прибыльные американские фильмы в домашнем прокате за последний год
CREATE VIEW USATopBoxofficeFilms AS
    SELECT FilmName, Budget, Income AS Boxoffice, (Income).Qnt - (Budget).Qnt
AS Profit
    FROM Film F NATURAL JOIN
        (SELECT FilmId, Income From Boxoffice B WHERE B.CountryId = 1) AS ABO
    WHERE
        F.FType = 'film' AND
        (F.Budget).Currency = 'USD' AND
        NOW() - F.ReleaseDate < INTERVAL '1 year' AND
        (FilmId, 1) IN (SELECT * FROM FilmCountry)
    ORDER BY Profit DESC LIMIT 100;

--- Список фильмов и сериалов, которые скоро должны выйти, их жанр и дата
выхода
CREATE VIEW SoonRelease AS
    SELECT FilmName, FType, Genres, ReleaseDate
    FROM
        (SELECT FilmId, FilmName, FType, ReleaseDate
        FROM Film
        WHERE
            ReleaseDate > now() AND
            ReleaseDate - now() < INTERVAL '6 month') AS S
        NATURAL JOIN
        (SELECT FilmId, STRING_AGG(concat(GenreName), ', ') AS Genres
        FROM FilmGenre NATURAL JOIN Genre
        GROUP BY FilmId) AS G
    ORDER BY ReleaseDate;

--- Сериалы, количество серий и продолжительность непрерывного просмотра
CREATE VIEW SeriesDuration AS
    SELECT FilmName, Series, (Series * Duration) AS TotalDuration
    FROM Film NATURAL JOIN
        (SELECT FilmId, COUNT(*) AS Series FROM Series GROUP BY FilmId) AS D
    WHERE
        FType = 'series';

--- Самые титулованные работники киноиндустрии
CREATE VIEW MostAwardFilmmakers AS
```

```
    SELECT FirstName || COALESCE(' ' || LastName, '') AS Name, Nominations,
Wins, (Nominations + Wins * 2) as Score
    FROM Person NATURAL JOIN
        (SELECT PersonId, Count(PNominationId) AS Nominations, Sum(CAST(Win
as Int)) AS Wins
        FROM PersonAward
        GROUP BY PersonId) AS W
    ORDER BY Score DESC;
```

**Роли**

```
--- Доступ на чтение - всем
GRANT SELECT
    ON TABLE Person, Country, Genre, Film, FilmGenre, FilmCountry, Score,
Series, Boxoffice,
     Profession, Filmmaker, AwardingOrg, FilmNomination, PersonNomination,
FilmAward, PersonAward
    TO Public;

GRANT SELECT
    ON TopFilms, USATopBoxofficeFilms, SoonRelease, SeriesDuration,
MostAwardFilmmakers
    TO Public;

--- Admin
--- Администратор ресурса имеет право изменять все данные
CREATE ROLE Admin;

GRANT SELECT, INSERT, UPDATE
    ON TABLE Person, Country, Genre, Film, FilmGenre, FilmCountry, Score,
Series, Boxoffice,
     Profession, Filmmaker, AwardingOrg, FilmNomination, PersonNomination,
FilmAward, PersonAward, Keys
    TO GROUP Admin;

GRANT EXECUTE
    ON FUNCTION
     CreatePerson(VARCHAR, VARCHAR, VARCHAR, DATE, VARCHAR),
     CreateFilm(VARCHAR, DATE, INTERVAL, FilmType, VARCHAR, MoneyType,
VARCHAR)
    TO GROUP Admin;

--- User
--- Пользователь ресурса. Может выставлять оценки
CREATE ROLE User;

GRANT INSERT, UPDATE
    ON TABLE Score
    TO GROUP User;

GRANT EXECUTE
    ON FUNCTION
        AddScore(INT, INT, FLOAT),
        BestPersonFilms(INT, FilmType)
    TO GROUP User;

--- Boxoffice Integrator
--- Интегратор для обновления данных boxoffice
CREATE ROLE BoxofficeIntegrator;
```

```
GRANT INSERT, UPDATE
    ON TABLE Score
    TO GROUP BoxofficeIntegrator;

GRANT EXECUTE
    ON FUNCTION UpdateBoxoffice(INT, INT, MoneyType)
    TO GROUP BoxofficeIntegrator;

--- Film Award
--- Представитель кино-премии. Может обновлять данные о наградах и номинациях
CREATE ROLE FilmAward;

GRANT INSERT, UPDATE
    ON TABLE FilmNomination, PersonNomination, FilmAward, PersonAward
    TO GROUP FilmAward;
```

**Функции и Триггеры**

```
--- Запрещяем добавлять информации о сериях для фильмов.
CREATE OR REPLACE FUNCTION CheckSerias()
    RETURNS TRIGGER AS $$
BEGIN
    IF NEW.FilmId IN (SELECT FilmId FROM Film WHERE FType = 'series') THEN
        RETURN NEW;
    ELSE
        RETURN NULL;
    END IF;
END;
$$ language plpgsql;

CREATE TRIGGER VerifyFilmType
BEFORE INSERT OR UPDATE ON Series
FOR EACH ROW EXECUTE PROCEDURE CheckSerias();

--- Запрещяем вручать профессиональные награды людям занимающиеся другим
родом деятельности
--- (нельзя вручать актерские награды операторам и наоборот)
CREATE OR REPLACE FUNCTION CheckAwardProfession()
    RETURNS TRIGGER AS $$
DECLARE
    AwardProfessionId INT;
BEGIN
    SELECT p.ProfessionId INTO AwardProfessionId
    FROM PersonNomination p
    WHERE p.PNominationId = NEW.PNominationId;
    IF AwardProfessionId IS NULL OR NEW.ProfessionId = AwardProfessionId THEN
        RETURN NEW;
    ELSE
        RETURN NULL;
    END IF;
END;
$$ language plpgsql;

CREATE TRIGGER VerifyAwardProfession
BEFORE INSERT OR UPDATE ON PersonAward
FOR EACH ROW EXECUTE PROCEDURE CheckAwardProfession();

--- Проверяем что победить в каждой номинации в рамках фестиваля один
CREATE OR REPLACE FUNCTION CheckOneFilmWinner()
    RETURNS TRIGGER AS $$
```

```
DECLARE
    CurrentWinners INT;
BEGIN
    IF NEW.Win = False THEN
        RETURN NEW;
    END IF;
    SELECT COUNT(*) INTO CurrentWinners
    FROM FilmAward FA
    WHERE FA.AwardId = NEW.AwardId AND FA.FNominationId = NEW.FNominationId
AND FA.Win = True;
    IF CurrentWinners = 0 THEN
        RETURN NEW;
    ELSE
        RETURN NULL;
    END IF;
END;
$$ language plpgsql;


CREATE TRIGGER VerifyOneFilmAwardWinner
BEFORE INSERT OR UPDATE ON FilmAward
FOR EACH ROW EXECUTE PROCEDURE CheckOneFilmWinner();

--- Тоже для личных наград
CREATE OR REPLACE FUNCTION CheckOnePersonWinner()
    RETURNS TRIGGER AS $$
DECLARE
    CurrentWinners INT;
BEGIN
    IF NEW.Win = False THEN
        RETURN NEW;
    END IF;
    SELECT COUNT(*) INTO CurrentWinners
    FROM PersonAward PA
    WHERE PA.AwardId = NEW.AwardId AND PA.PNominationId = NEW.PNominationId
AND PA.Win = True;
    IF CurrentWinners = 0 THEN
        RETURN NEW;
    ELSE
        RETURN NULL;
    END IF;
END;
$$ language plpgsql;


CREATE TRIGGER VerifyOnePersonAwardWinner
BEFORE INSERT OR UPDATE ON PersonAward
FOR EACH ROW EXECUTE PROCEDURE CheckOnePersonWinner();


--- Получить текущую оценку фильма
CREATE OR REPLACE FUNCTION CurrentFilmScore(FId INT) RETURNS Table(FilmScore
FLOAT)
AS $$
BEGIN
    RETURN QUERY
    SELECT AVG(Score) AS FilmScore
    FROM Score S
    WHERE S.FilmId = FId;
```

```
END;
$$ language plpgsql;

--- Получить следующий ключ для таблицы
CREATE OR REPLACE FUNCTION NextKey(KName VARCHAR(50)) RETURNS INT
AS $$
DECLARE
    NextKey INT;
BEGIN
    SELECT MinKey INTO NextKey FROM Keys WHERE Keys.TableName = KName;
    UPDATE Keys SET MinKey = MinKey + 1 WHERE Keys.TableName = KName;
    RETURN NextKey;
END;
$$ language plpgsql;

--- Создать профиль человека в системе
CREATE OR REPLACE FUNCTION CreatePerson(
    FName VARCHAR(100),
    LName VARCHAR(100),
    PMic VARCHAR(100),
    Birthdate DATE,
    PCountry VARCHAR(100))
RETURNS INT
AS $$
DECLARE
    PId INT;
    CId INT;
BEGIN
    SELECT * FROM NextKey('Person') INTO PId;
    SELECT CountryId INTO CId FROM Country WHERE Country.CountryName =
PCountry;
    INSERT INTO Person
        (PersonId, FirstName,LastName,Birthday,Patronymic, CountryId)
    VALUES
        (PId, FName, LName, Birthdate, PMic, CId);
    RETURN PId;
END;
$$ language plpgsql;

--- Создать профиль фильма в системе
CREATE OR REPLACE FUNCTION CreateFilm(
    mFilmName VARCHAR(100),
    mReleaseDate DATE,
    mDuration INTERVAL,
    mFType FilmType,
    mSlogan VARCHAR(200),
    mBudget MoneyType,
    mDescription VARCHAR(1000))
RETURNS INT
AS $$
DECLARE
    FId INT;
BEGIN
    SELECT * FROM NextKey('Film') INTO FId;
    INSERT INTO Film
        (FilmId,FilmName,ReleaseDate,Duration,FType,Slogan,Budget,
Description)
    VALUES
```

```
        (FId, mFilmName, mReleaseDate, mDescription, mFType, mSlogan,
mBudget, mDescription);
    RETURN FId;
END;
$$ language plpgsql;

--- Добавить новую информацию о сборах.
--- Возвращяет суммарное заработанное количество денег фильмом в конкретной
стране или NULL
--- если пытаемся увеличить сборы в друго валюте
CREATE OR REPLACE FUNCTION UpdateBoxoffice(mFilmId INT,  mCountryId INT,
newBoxOffice MoneyType)
RETURNS MoneyType
AS $$
DECLARE
    CurrentBoxoffice MoneyType;
BEGIN
    SELECT (Income).Qnt, (Income).Currency INTO CurrentBoxoffice
    FROM Boxoffice
    WHERE FilmId = mFilmId AND CountryId = mCountryId;

    IF NOT FOUND THEN
        INSERT INTO Boxoffice(FilmId, CountryId, Income) VALUES (mFilmId,
mCountryId, newBoxOffice);
        RETURN newBoxOffice;
    END IF;

    IF CurrentBoxoffice.Currency = newBoxOffice.Currency THEN
        UPDATE Boxoffice SET Income.Qnt = (Income).Qnt + (newBoxOffice).Qnt
        WHERE FilmId = mFilmId AND CountryId = mCountryId;
        RETURN ((CurrentBoxoffice).Qnt + (newBoxOffice).Qnt,
newBoxOffice.Currency);
    ELSE
        RETURN NULL;
    END IF;
END;
$$ language plpgsql;

--- Добавить оценку фильму
CREATE OR REPLACE FUNCTION AddScore(mUserId INT, mFilmId INT, mScore FLOAT)
RETURNS VOID
AS $$
BEGIN
    INSERT INTO Score(UserId, FilmId, Score) VALUES (mUserId, mFilmId,
mScore)
    ON CONFLICT (UserId, FilmId) DO UPDATE SET Score = mScore;
END;
$$ language plpgsql;

--- Самые лучшие фильмы или сериалы в которых человек принимал участие с
оценкой выше 7
CREATE OR REPLACE FUNCTION BestPersonFilms(mPersonId INT, mFilmType FilmType)
RETURNS Table(FilmName VARCHAR(100), ProfessionName VARCHAR(100), FilmScore
FLOAT)
AS $$
BEGIN
    RETURN QUERY
    SELECT Film.FilmName, Profession.ProfessionName, FS.FilmScore FROM
     (SELECT FilmId, ProfessionId FROM Film NATURAL JOIN
```

```
        (SELECT FilmId, ProfessionId
        FROM Filmmaker
        WHERE PersonId = mPersonId) AS PF
    WHERE FType = mFilmType)AS FN
    NATURAL JOIN
    (SELECT FilmId, AVG(Score) AS FilmScore
     FROM Score
     GROUP BY FilmId) AS FS
    NATURAL JOIN Film NATURAL JOIN Profession
    WHERE FS.FilmScore > 7
    ORDER BY FS.FilmScore DESC;
END;
$$ language plpgsql;
```