

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ  
УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ,  
МЕХАНИКИ И ОПТИКИ**

Факультет: Программной инженерии и компьютерной техники  
Кафедра: Информатики и прикладной математики  
Направление подготовки: 09.03.04 (Программная инженерия)

Дисциплина «Базы Данных»  
**Курсовая работа**

Группа: Р3117

**Работу выполнили**  
Минин Александр  
Молодецкий Арсений

2018г.

# Предметная область

(оригинальная версия от заказчика с заметками)

В качестве предметной области будет рассмотрена система кинотеатров в городе Санкт-Петербург. База данных будет представлять собой некоторое состояние кинотеатров в определенный период времени. Целью такого описания является составление бд, по которой бы было удобно определять наиболее релевантные варианты похода в кино, которые можно оформить в виде информационного ресурса. Мы хотим создать ресурс, помогающий пользователям выбрать интересующий их фильм. По актуальности можно отметить, что это одно из наиболее посещаемых видов развлечений под крышей, особенно в таком пасмурном городе.

Как всем известно, в кинотеатрах показывают фильмы во время сеансов в кинозалах. Фильмы это продукты киноискусства, которое зародилось недавно. Каждая кинокартина имеет длительность показа (иначе говоря, продолжительность сеанса) и жанр (чаще всего фильм относится сразу к нескольким). Стоит также отметить, что кинолента имеет год выпуска и место съемки. Современная публика достаточно требовательная, поэтому ей необходимо больше разнообразия пейзажа на экране. Для этого кадры снимают в разных частях света, разные студии на множестве различных съёмочных площадок. В съемках также принимают участие различные актеры, операторы, сценаристы, режиссеры, продюсеры, композиторы, художники, монтажеры и многие другие (съёмочная группа)<sup>1</sup>. Аренда съёмочной площадки, перелеты группы да и сама оплата работы всего персонала требует колоссального количества денежных средств. Эти деньги должны укладываться в бюджет фильма (в теории). В конце работы над фильмом в ход идут маркетологи. Они придумывают броский слоган, призывающий людей посмотреть фильм (желательно в месте проката или на DVD), тем самым подогревая интерес зрителей и повышая рейтинг ожидаемости. Ко всему прочему, данные специалисты назначают даты премьеры и показа, продумывают разнообразный медиа контент (саундтреки, постеры, афиши, трейлеры, рекламу). Чаще всего слоган отражает возрастной рейтинг кинокартины (G — General audiences, PG — Parental guidance suggested, PG-13 — Parents strongly cautioned, R — Restricted, NC-17 — No One 17 & Under Admitted), но назначается он в зависимости от содержания фильма. После показа кинокритики выставляют рейтинги фильму и пишут отзывы. В случае приложения на смартфоне 1 пользователь сможет поставить оценку от 1 до 10 для каждого фильма 1 раз, а в случае сайта можно будет выставлять оценку неограниченное количество раз и по этим оценкам тоже выстраивается рейтинг фильмов<sup>2</sup>. ~~После того, как фильм закончился в прокате, появляются данные о кассовых сборах по всему миру и происходит релиз на DVD и Blu-ray.~~ Также фильмы могут быть номинированы или премированы на разных церемониях всякими наградами, что тоже может быть интересно зрителю. И наконец если фильм понравился пользователю, он может посмотреть список похожих фильмов (схожесть по жанрам, оценкам, съёмочной группе, студии и т.д.).

Теперь рассмотрим систему кинотеатров. Она представляет собой совокупность всех сетей и частных кинотеатров (кинотеатры, которые не входят в сети кинотеатров), находящихся в черте города. У каждого кинотеатра есть физический адрес, список сеансов в календаре событий. Также в кинотеатрах есть залы, в которых проходят сеансы, во время которых показываются определенные фильмы в определенные даты и время. Сеансы имеют длительность и цену билета, которая зачастую зависит от времени суток, дня недели. а также расположение места в кинозале<sup>3</sup>. Места в кинозале пронумерованы и сгруппированы по рядам, что позволяет посетителю без

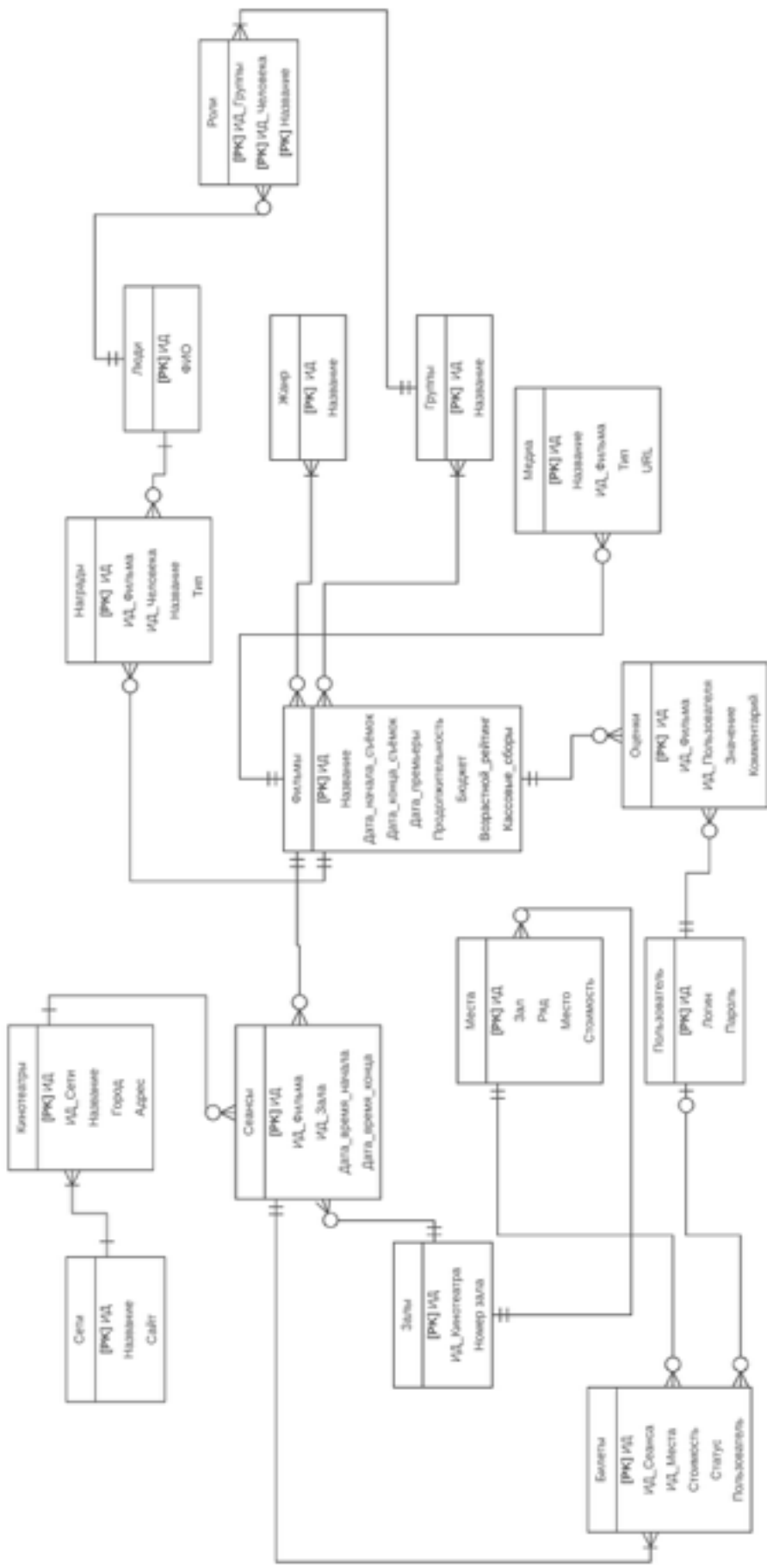
особого труда найти подходящее место или посмотреть занятость зала, ведь места могут быть свободны, куплены или забронированы.

Таким образом, благодаря нашей системе, можно, не теряя времени, узнать информацию об интересующем нас фильме, а также найти удобное место для просмотра киноленты. Хотим отметить, что наша система может способствовать привлечению потока клиентов в кинотеатры, так как пользователи смогут оптимально выбирать релевантный сеанс.

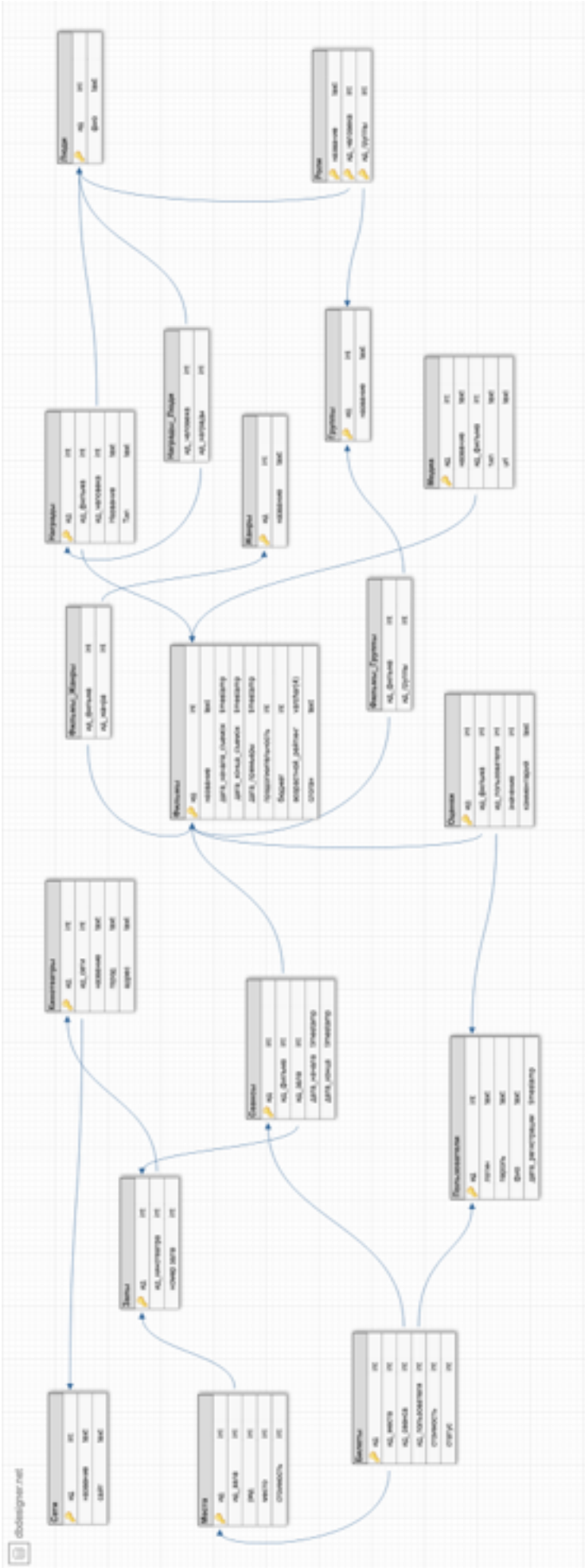
В ходе обсуждения проекта с заказчиком были уточнены детали:

- 1) Над одним фильмом может работать несколько групп (команд), состоящих из работников киноиндустрии: режиссера, оператора и так далее. Каждый человек в группе имеет одну или более должностей (ролей).
- 2) Данные о релизах хранить не нужно. Пользователь сможет выставить оценку для фильма только один раз: и на смартфоне, и на сайте.
- 3) Следует хранить расположение мест в зале и “базовую” стоимость (тариф) для каждого места, которую можно менять для каждого отдельно взятого билета/сеанса. В терминах инфологической модели БД, у нас должны быть атрибуты “стоимость” и в таблице, предназначенной для хранения билетов, и в таблице, предназначенной для хранения мест и их конфигурации.

# Инфологическая модель



# Даталогическая модель



```
CREATE TABLE "Сети" (  
    "ид" SERIAL NOT NULL,  
    "название" TEXT NOT NULL UNIQUE,  
    "url" TEXT UNIQUE,  
    PRIMARY KEY ("ид")  
);
```

```
CREATE TABLE "Кинотеатры" (  
    "ид" SERIAL NOT NULL,  
    "ид_сети" INTEGER NOT NULL REFERENCES "Сети" ON DELETE CASCADE,  
    "название" TEXT NOT NULL,  
    "город" TEXT NOT NULL,  
    "адрес" TEXT NOT NULL,  
    PRIMARY KEY ("ид")  
);
```

```
CREATE TABLE "Залы" (  
    "ид" SERIAL NOT NULL,  
    "ид_кинотеатра" INTEGER NOT NULL REFERENCES "Кинотеатры" ON DELETE  
CASCADE,  
    "номер_зала" int NOT NULL CONSTRAINT csr_room_id CHECK ("номер_зала" > 0),  
    PRIMARY KEY ("ид")  
);
```

```
CREATE TABLE "Места" (  
    "ид" SERIAL NOT NULL,  
    "ид_зала" INTEGER NOT NULL REFERENCES "Залы" ON DELETE CASCADE,  
    "ряд" int NOT NULL  
        CONSTRAINT csr_row CHECK ("ряд" > 0),  
    "место" int NOT NULL  
        CONSTRAINT csr_place CHECK ("место" > 0),  
    "стоимость" int NOT NULL  
        CONSTRAINT csr_seat_price CHECK ("стоимость" > 0),  
    PRIMARY KEY ("ид")  
);
```

```
CREATE TABLE "Жанры" (  
    "ид" SERIAL NOT NULL,  
    "название" TEXT NOT NULL UNIQUE,  
    PRIMARY KEY ("ид")  
);
```

```
CREATE TABLE "Люди" (  
    "ид" SERIAL NOT NULL,  
    "фio" TEXT NOT NULL,  
    PRIMARY KEY ("ид")  
);
```

```
CREATE TABLE "Пользователи" (  
    "ид" SERIAL NOT NULL,  
    "логин" TEXT NOT NULL UNIQUE,
```

```

        "пароль" TEXT NOT NULL,
        "фио" TEXT NOT NULL,
        "дата_регистрации" TIMESTAMP NOT NULL
        CONSTRAINT csr_date_reg CHECK ("дата_регистрации" <=
NOW()),
        PRIMARY KEY ("ид")
);

```

```

CREATE TABLE "Фильмы" (
    "ид" SERIAL NOT NULL,
    "название" TEXT NOT NULL,
    "дата_начала_съемок" TIMESTAMP NOT NULL,
    "дата_конца_съемок" TIMESTAMP,
    "дата_премьеры" TIMESTAMP,
    "продолжительность" int NOT NULL
        CONSTRAINT csr_duration CHECK ("продолжительность" > 0),
    "бюджет" int,
        CONSTRAINT csr_budget CHECK ("бюджет" > 0),
    "возрастной_рейтинг" VARCHAR(4) NOT NULL CHECK("возрастной_рейтинг"
in('G','PG','PG-13','R','NC-17')),
    "слоган" TEXT,
        CONSTRAINT csr_movie_start_end_range CHECK ("дата_конца_съемок" >
"дата_начала_съемок"),
        CONSTRAINT csr_movie_release_end CHECK ("дата_конца_съемок" <
"дата_премьеры"),
    PRIMARY KEY ("ид")
);

```

```

CREATE TABLE "Медиа" (
    "ид" SERIAL NOT NULL,
    "название" TEXT NOT NULL,
    "ид_фильма" INTEGER NOT NULL REFERENCES "Фильмы" ON DELETE
CASCADE,
    "тип" TEXT NOT NULL,
    "url" TEXT NOT NULL,
    PRIMARY KEY ("ид")
);

```

```

CREATE TABLE "Оценки" (
    "ид_фильма" INTEGER NOT NULL REFERENCES "Фильмы" ON DELETE
CASCADE,
    "ид_пользователя" INTEGER NOT NULL REFERENCES "Пользователи" ON
DELETE CASCADE,
    "значение" int NOT NULL CONSTRAINT csr_rate CHECK ("значение" >= 1 AND
"значение" <= 10),
    "комментарий" TEXT,
    PRIMARY KEY ("ид_фильма", "ид_пользователя")
);

```

```

CREATE TABLE "Награды" (
    "ид" SERIAL NOT NULL,

```

```

        "ид_фильма" INTEGER NOT NULL REFERENCES "Фильмы" ON DELETE
CASCADE,
        "ид_человека" INTEGER NOT NULL REFERENCES "Люди" ON DELETE CASCADE,
        "Название" TEXT NOT NULL,
        "Тип" TEXT NOT NULL,
        PRIMARY KEY ("ид")
);

CREATE TABLE "Группы" (
    "ид" SERIAL NOT NULL,
    "название" TEXT NOT NULL,
    PRIMARY KEY ("ид")
);

CREATE TABLE "Роли" (
    "название" TEXT NOT NULL,
    "ид_человека" INTEGER NOT NULL REFERENCES "Люди" ON DELETE CASCADE,
    "ид_группы" INTEGER NOT NULL REFERENCES "Группы" ON DELETE CASCADE,
    PRIMARY KEY ("название", "ид_человека", "ид_группы")
);

CREATE TABLE "Сеансы" (
    "ид" SERIAL NOT NULL,
    "ид_фильма" INTEGER NOT NULL REFERENCES "Фильмы" ON DELETE
CASCADE,
    "ид_зала" INTEGER NOT NULL REFERENCES "Залы" ON DELETE CASCADE,
    "дата_начала" TIMESTAMP NOT NULL,
    "дата_конца" TIMESTAMP NOT NULL,
    CONSTRAINT csr_session_date CHECK ("дата_начала" < "дата_конца"),
    PRIMARY KEY ("ид")
);

CREATE TABLE "Билеты" (
    "ид_места" INTEGER NOT NULL REFERENCES "Места" ON DELETE RESTRICT,
    "ид_сеанса" INTEGER NOT NULL REFERENCES "Сеансы" ON DELETE CASCADE,
    "ид_пользователя" int REFERENCES "Пользователи" ON DELETE RESTRICT,
    "стоимость" int NOT NULL CONSTRAINT csr_ticket_price CHECK ("стоимость" > 0),
    "статус" int NOT NULL,
    PRIMARY KEY ("ид_места", "ид_сеанса")
);

CREATE TABLE "Фильмы_Жанры" (
    "ид_фильма" INTEGER NOT NULL REFERENCES "Фильмы" ON DELETE
CASCADE,
    "ид_жанра" INTEGER NOT NULL REFERENCES "Жанры" ON DELETE CASCADE,
    PRIMARY KEY ("ид_фильма", "ид_жанра")
);

CREATE TABLE "Фильмы_Группы" (

```



```

        "ид_фильма" INTEGER NOT NULL REFERENCES "Фильмы" ON DELETE
CASCADE,
        "ид_группы" INTEGER NOT NULL REFERENCES "Группы" ON DELETE CASCADE,
        PRIMARY KEY ("ид_фильма", "ид_группы")
);

CREATE TABLE "Награды_Люди" (
        "ид_человека" INTEGER NOT NULL REFERENCES "Люди" ON DELETE CASCADE,
        "ид_награды" INTEGER NOT NULL REFERENCES "Награды" ON DELETE
CASCADE,
        PRIMARY KEY ("ид_человека", "ид_награды")
);

```

## Триггеры

```

CREATE OR REPLACE FUNCTION сеансы_до_премьеры_запрещены () RETURNS trigger
AS $сеансы_до_премьеры_запрещены$
DECLARE
        дата_премьеры timestamp;
BEGIN
SELECT Фильмы.дата_премьеры INTO дата_премьеры FROM "Фильмы" WHERE ид =
NEW.ид_фильма;
IF NEW.дата_начала < дата_премьеры THEN
        RAISE EXCEPTION 'Дата премьеры фильма (%) не может быть позже, чем дата
начала показа (%)', дата_премьеры, NEW.дата_начала;
END IF;

RETURN NEW;
END;
$сеансы_до_премьеры_запрещены$ LANGUAGE plpgsql;

CREATE TRIGGER "сеансы_до_премьеры_запрещены" BEFORE INSERT OR UPDATE
ON "Сеансы"
FOR EACH ROW EXECUTE PROCEDURE сеансы_до_премьеры_запрещены();

CREATE OR REPLACE FUNCTION награды_до_премьеры_запрещены() RETURNS
trigger AS $награды_до_премьеры_запрещены$
DECLARE
        дата_премьеры timestamp;
BEGIN
SELECT Фильмы.дата_премьеры INTO дата_премьеры FROM "Фильмы" WHERE ид =
NEW.ид_фильма;
IF now() < дата_премьеры THEN
        RAISE EXCEPTION 'Награда не может вручаться до выхода фильма (преьера
%)', дата_премьеры;
END IF;
RETURN NEW;
END;
$награды_до_премьеры_запрещены$ LANGUAGE plpgsql;

```

```
CREATE TRIGGER "награды_до_премьеры_запрещены" BEFORE INSERT OR UPDATE  
ON "Награды"  
FOR EACH ROW EXECUTE PROCEDURE награды_до_премьеры_запрещены();
```

## Типичные запросы

Типичные (распространенные) запросы к СУБД составлены на основании предметной области и предполагаемых моделей использования БД. Здесь приведены самые распространенные запросы, которые позволяют нам понять, какие индексы нужно ввести в первую очередь, чтобы СУБД как можно более эффективно справлялась с выполнением требований заказчика.

- Для вывода оценок за фильмы у пользователя и любимых жанров

```
SELECT * FROM Оценки JOIN Фильмы ON Фильмы.ид = Оценки.ид_фильма WHERE  
Оценки.ид_пользователя = ?;  
SELECT * FROM Оценки JOIN Фильмы_Жанры ON Фильмы_Жанры.ид_фильма =  
Оценки.ид_фильма;
```

- Для вывода оценок за фильм

```
SELECT * FROM Оценки JOIN Фильмы ON Фильмы.ид = Оценки.ид_фильма WHERE  
Оценки.ид_фильма = ?;
```

- Для сравнения вкусов к фильмам у двух пользователей

```
SELECT Фильмы.название, Оценки.значение AS Оценка_1ого, t.значение AS  
Оценка_2ого  
FROM Оценки  
JOIN Фильмы ON Фильмы.ид = Оценки.ид_фильма  
JOIN Оценки AS t ON t.ид_фильма = Оценки.ид_фильма AND t.ид_пользователя !=  
Оценки.ид_пользователя  
WHERE Оценки.ид_пользователя = ? AND t.ид_пользователя = ?;
```

- Для нахождения пароля определенного пользователя

```
SELECT Пароль FROM Пользователи WHERE логин = ?;
```

- Для вывода информации сеансах на фильмы, идущие в данный момент

```
SELECT * FROM Фильмы  
JOIN Сеансы  
WHERE Фильмы.ИД = Сеансы.ИД  
AND Фильмы.Дата_премьеры - CURR_DATE < ~ 2 НЕДЕЛИ  
AND (Дата_время_начала < ? OR Дата_время_конца < ?);
```

- Для вывода информации о фильме: в частности о тех, кто в нем снимался и в какой должности (роли)

```
SELECT * FROM Фильмы  
JOIN Группы ON Фильмы_Группы.ид_фильма = Фильмы.ид  
JOIN Роли ON Роли.ид_группы = Фильмы_Группы.ид_группы  
JOIN Люди ON Роли.ид_человека = Люди.ид  
WHERE Фильмы.ид=?;
```

- Для вывода информации о конкретном сеансе и билетах

```
SELECT * FROM Сеансы
```

```
JOIN Залы ON Сеансы.ид_зала = Залы.ид  
JOIN Места ON Места.ид_зала = Залы.ид  
LEFT JOIN Билеты ON Билеты.ид_сеанса = Сеансы.ид AND Билеты.ид_места =  
Места.ид  
WHERE Сеансы.ид = ?;
```

## **Индексы**