

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ
по дисциплине «Базы данных»
Вариант 10

Студент гр. 8382

Терехов А.Е.

Преподаватель

Фомичева Т.Г.

Санкт-Петербург

2020

СОДЕРЖАНИЕ

1 Постановка задачи.....	5
1.1 Задание.....	5
2 Проектирование БД.....	7
2.1 ER-модель.....	7
2.2 Реляционная модель.....	9
3 Структуры таблиц и постоянные связи между таблицами.....	16
3.1 Структуры таблиц.....	16
3.2 Постоянные связи между таблицами.....	18
4 Содержимое таблиц.....	19
5 Схема иерархии интерфейса.....	23
6 Описание окон.....	24
6.1 Главная (Root).....	24
6.2 Кинотеатр – Кинотеатры (CinemaWindow).....	25
6.3 Кинотеатр – Сеансы (CinemaSessions).....	26
6.4 Кинотеатр – Афиши (PostersWindow).....	27
6.5 Кинотеатр – Добавление зала или сеанса (CinemaInsertSession).....	29
6.6 Справочная служба – Главное окно (FaqWindow).....	30
6.7 Справочная служба – Кинотеатры (CinemaWindow).....	31
6.8 Справочная служба – Сеансы (CinemaSessions).....	32
6.9 Справочная служба – Афиши (PostersWindow).....	32
6.10 Справочная служба – Добавление зала или сеанса (CinemaInsertSession)	32
6.11 Справочная служба – Перечень фильмов (FilmsWindow).....	33
6.12 Справочная служба – Роли фильма (FilmsActors).....	34
6.13 Справочная служба – Добавление роли (FilmsInsertRole).....	35
6.14 Справочная служба – Призы фильма (FilmsPrizes).....	35

6.15	Справочная служба – Добавление приза (FilmsInsertPrize).....	36
6.16	Справочная служба – Постер фильма (FilmPoster).....	36
6.17	Справочная служба – Сеансы фильма.....	37
6.18	Справочная служба – Перечень актеров.....	37
6.19	Справочная служба – Сеансы при участии заданного актера.....	38
6.20	Справочная служба – Поиск.....	39
6.21	Справочная служба – Отчет.....	41
7	Описание запросов.....	44
7.1	Обновление кинотеатра.....	44
7.2	Обновление сеанса.....	44
7.3	Обновление фильма.....	44
7.4	Обновление актера.....	45
7.5	Вставка кинотеатра.....	45
7.6	Вставка зала.....	45
7.7	Вставка фильма.....	45
7.8	Вставка сеанса.....	45
7.9	Вставка жанра.....	45
7.10	Вставка роли.....	45
7.11	Вставка приза.....	46
7.12	Удаление кинотеатра.....	46
7.13	Удаление фильма.....	46
7.14	Удаление сеанса.....	46
7.15	Удаление роли.....	46
7.16	Удаление актера.....	46
7.17	Удаление награды.....	47
	Заключение.....	48
	ПРИЛОЖЕНИЕ А.....	49

ПРИЛОЖЕНИЕ Б.....	51
ПРИЛОЖЕНИЕ В.....	55

1 Постановка задачи

1.1 Задание.

Пусть требуется создать программную систему, предназначенную для работников справочной службы кинотеатров города. Такая система должна обеспечивать хранение сведений о кинотеатрах города, о фильмах, которые в них демонстрируются, о сеансах и билетах на эти сеансы.

Сведения о кинотеатре - это его название, район города, где расположен кинотеатр, категория, вместимость.

Сведения о фильме - это название фильма, режиссер, оператор, актеры, сыгравшие главные роли, жанр; производство, наличие призов кинофестивалей, продолжительность сеанса, кадр из фильма для рекламы.

Кроме того, должна храниться информация о репертуаре кинотеатров на месяц, то есть о том какие фильмы, когда и где демонстрируются, о ценах на билеты и о количестве свободных мест на тот или иной сеанс. На разных сеансах в одном кинотеатре могут идти разные фильмы, а если в кинотеатре несколько залов, то и на одном.

Кинотеатр может ввести новый фильм в репертуар или убрать фильм из репертуара. Работник справочной службы может корректировать перечень фильмов, находящихся в прокате – добавлять новые фильмы и снимать с проката, а также перечень кинотеатров, поскольку кинотеатры могут открываться или закрываться, причем иногда временно, например, на ремонт. Цена билета определяется прокатной стоимостью копии фильма, сеансом и категорией кинотеатра.

Справочной службе могут потребоваться следующие сведения о текущем состоянии проката фильмов в городе:

- Репертуар кинотеатра?
- Адрес и район кинотеатра ?

- Число свободных мест на данный сеанс в указанном кинотеатре?
- Цена билетов на данный сеанс в указанном кинотеатре?
- Жанр, производство и режиссер данного фильма ?
- Какие фильмы имеют награды, когда и в каких кинотеатрах они демонстрируются?
- В каких кинотеатрах в указанный день на указанных сеансах демонстрируется комедия?
- В каких кинотеатрах и когда демонстрируются фильмы с участием указанного актера?

Необходимо предусмотреть возможность создания афиши для кинотеатра, в которую будут помещены все имеющиеся в базе сведения о фильме, включая кадр из фильма, а также сведения о том, на каких сеансах этот фильм демонстрируется в указанном кинотеатре.

Сотрудники справочной службы должны также иметь возможность получить сгруппированный по районам города отчет за прошедший месяц о прокате фильмов (сколько и какие фильмы, в каких кинотеатрах демонстрировались, средняя цена билета на эти фильмы в каждом кинотеатре, доход по каждому кинотеатру и по району в целом).

В отчете также должно быть подсчитано, сколько всего фильмов находилось в прокате, сколько из них относятся к каждому из жанров, каков суммарный доход кинотеатров города за вычетом прокатной стоимости копий.

2 Проектирование БД

Для проектирования базы данных выбран ER-метод (метод "сущность-связь").

При использовании этого метода необходимо прежде всего создать ER-модель, отражающую связи сущностей заданной предметной области.

Описание сущностей включает в себя перечисление атрибутов сущностей – их свойств, необходимых для решения задачи, один или несколько атрибутов могут быть ключевыми, то есть однозначно определяющими экземпляр сущности.

2.1 ER-модель

Сущность Кинотеатр имеет три атрибута: его название, адрес и район. Каждый экземпляр однозначно идентифицируется названием и адресом.

Сущность Зал имеет три атрибута: номер, вместимость и категория. Каждый экземпляр однозначно идентифицируется номером.

Сущность Актер имеет три атрибута: имя, фамилия и дата рождения. Каждый экземпляр однозначно идентифицируется всеми тремя атрибутами одновременно.

Сущность Приз имеет три атрибута: название награды, год вручения этой награды и номинация. Каждый экземпляр однозначно идентифицируется всеми тремя атрибутами одновременно.

Сущность Фильм имеет семь атрибутов: название фильма, режиссер, оператор, стоимость проката, страна, длительность и кадр для афиши. Каждый экземпляр однозначно идентифицируется всеми названием и режиссером.

Сущность Жанр имеет один атрибут: название жанра. Каждый экземпляр однозначно идентифицируется своим единственным атрибутом.

Сущность Зал зависит от сущности Кинотеатр, поскольку зал не может существовать отдельно от кинотеатра.

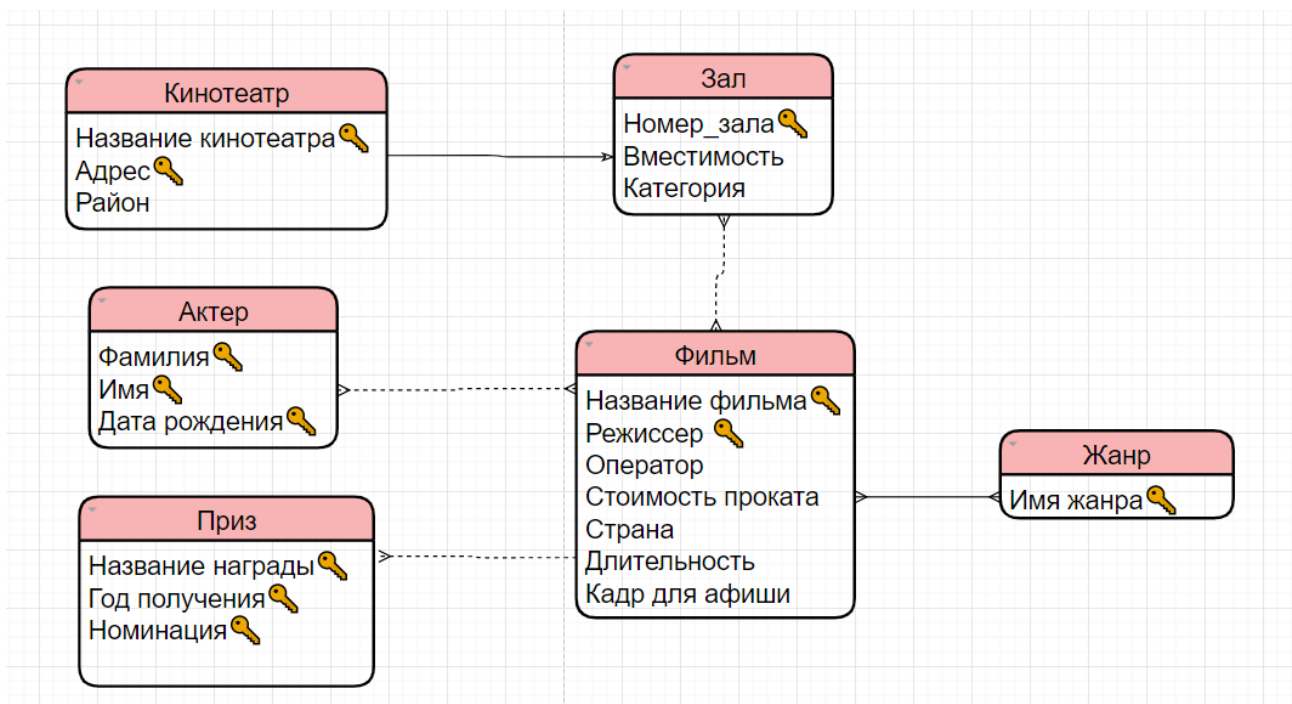
Связь между сущностями Фильм и Зал является необязательной связью многие-ко-многим, поскольку в каждом зале могут показываться разные фильмы, а

фильм может показываться в разных залах, но зал может быть закрыт и фильмы в нем показываться не будут, а фильм может выйти из проката.

Связь между сущностями Фильм и Актер является необязательной связью многие-ко-многим, поскольку в фильме могут не принимать участие никакие актеры, например, документальные фильмы или наоборот в фильме участвуют много актеров, а актер может не сниматься в фильмах или наоборот принимать участие в множестве фильмов.

Связь между сущностями Фильм и Приз является необязательной связью один-ко-многим, поскольку фильм может иметь несколько наград или не иметь их, а каждая награда обязательно принадлежит только одному фильму или не принадлежит ни одному из представленных фильмов.

Связь между сущностями Фильм и Жанр является обязательной связью многие-ко-многим, поскольку существует множество фильмов принадлежащих нескольким жанрам и у одного жанра может быть несколько фильмов.



Нераспределенные атрибуты: Дата, Время (сеанса), Цена билета, Количество свободных мест, Кого играет (актер).

Составив ER-модель, можем перейти к созданию реляционной модели базы данных.

2.2 Реляционная модель

По правилу 7 – связь Кинотеатр – Зал образует два отношения – по одному для каждой сущности со своими ключами. В отношение Зал добавляются ключевые атрибуты сущности Кинотеатр.

Получаем для отношения Кинотеатр составной ключ (Название кинотеатра, Адрес).

Получаем для отношения Зал составной ключ (Название кинотеатра, Адрес, Номер зала).

По правилу 6 – связь Зал – Фильм образует три отношения – два объектных и одно связное. Связное отношение Сеанс содержит ключевые атрибуты сущностей Зал и Фильм.

Получаем для отношения Зал составной ключ (Название кинотеатра, Адрес, Номер зала).

Получаем для отношения Фильм составной ключ (Название фильма, Режиссер).

Ключ связного отношения Сеанс будет состоять из следующих атрибутов: Название кинотеатра, Адрес, Номер зала, Дата, Время.

По правилу 6 – связь Фильм – Актер так же образует три отношения – два объектных и одно связное.

Получаем для отношения Фильм составной ключ (Название фильма, Режиссер).

Получаем для отношения Актер составной ключ (Имя, Фамилия, Дата рождения).

Связное отношение Роль содержит ключевые атрибуты сущностей Фильм и Актер: Название фильма, Режиссер, Имя, Фамилия, Дата рождения.

Так как отношение Актер избыточно – полностью содержится в связном отношении – оно создаваться не будет.

Для связи Фильм – Приз нужно создать три отношения два объектных и одно связное – Приз показываемого фильма, содержащее ключевые атрибуты обеих сущностей: Название фильма, Режиссер, Название награды, Год получения, Номинация.

Получаем для отношения Фильм составной ключ (Название фильма, Режиссер).

Получаем для отношения Приз составной ключ (Название награды, Год получения, Номинация). Так как отношение Приз избыточно – полностью содержится в связном отношении, а награды фильмов, которые не показываются в кинотеатрах, не интересны, оно создаваться не будет.

Для связи Фильм – Жанр создано три отношения два объектных и одно связное – Жанр фильма, содержащее ключевые атрибуты обеих сущностей: Название фильма, Режиссер, Имя жанра.

Получаем для отношения Фильм составной ключ (Название фильма, Режиссер).

Получаем для отношения Жанр ключ (Имя жанра). Так как отношение Жанр избыточно – полностью содержится в связном отношении, оно создаваться не будет.

Рассмотрим полученные отношения:

1) Кинотеатр (Название кинотеатра, Адрес, Район)

Функциональные зависимости:

Название кинотеатра, Адрес → Район – является ключом.

Не являются ФЗ:

Название кинотеатра → Адрес, Район – т.к. в городе может быть несколько кинотеатров с одним названием.

Название кинотеатра, Район → Адрес – т.к. в одном районе может быть несколько кинотеатров с одним названием.

Адрес → Название кинотеатра, Район – т.к. по одному адресу может быть несколько кинотеатров.

Отношение Кинотеатр находится в НФБК, т.к. все его детерминанты являются возможными ключами.

2) Зал (Название кинотеатра, Адрес, Номер зала, Вместимость, Категория)

Функциональные зависимости:

Название кинотеатра, Адрес, Номер зала → Вместимость, Категория – является ключом.

Не являются ФЗ:

Название кинотеатра, Адрес → Номер зала, Вместимость, Категория – т.к. в одном кинотеатре может быть несколько залов.

Название кинотеатра, Адрес, Вместимость → № зала, Категория – т.к. в одном кинотеатре могут существовать залы одинаковой вместимости.

Номер зала → Название кинотеатра, Адрес, Вместимость, Категория – т.к. в разных кинотеатрах могут быть залы с одинаковыми номерами.

Категория → Название кинотеатра, Адрес, Номер зала, Вместимость – т.к. в разных кинотеатрах могут быть залы одинаковой категории.

Отношение Зал находится в НФБК, т.к. все его детерминанты являются возможными ключами.

3) Фильм (Название фильма, Режиссер, Оператор, Стоимость проката, Страна, Длительность, Кадр для афиши).

Функциональные зависимости:

Название фильма, Режиссер → Оператор, Стоимость проката, Страна, Длительность, Кадр для афиши – является ключом

Название фильма, Оператор → Режиссер, Стоимость проката, Страна, Длительность, Кадр для афиши – является возможным ключом.

Избыточные ФЗ:

Название фильма, Режиссер, Оператор → Стоимость проката, Страна, Длительность, Кадр для афиши – получается из приведенных выше ФЗ с помощью добавления соответствующего атрибута.

Не являются ФЗ:

Название фильма, Страна → Оператор, Режиссер, Стоимость проката, Длительность, Кадр для афиши – т.к. в одной стране может быть снято несколько фильмов с одинаковым названием.

Режиссер → Название фильма, Оператор, Стоимость проката, Страна, Длительность, Кадр для афиши – т.к. режиссер может участвовать в съемках разных фильмов.

Оператор → Название фильма, Режиссер, Стоимость проката, Страна, Длительность, Кадр для афиши – т.к. оператор может участвовать в съемках разных фильмов.

Отношение Фильм находится в НФБК, т.к. все его детерминанты являются возможными ключами.

4) Сеанс (Название кинотеатра, Адрес, Номер зала, Дата, Время, Название фильма, Режиссер, Цена билета, Количество свободных мест)

Функциональные зависимости:

Название кинотеатра, Адрес, Номер зала, Дата, Время → Название фильма, Режиссер, Цена билета, Количество свободных мест – является ключом.

Избыточные ФЗ:

Название кинотеатра, Адрес, Номер зала, Дата, Время, Название фильма, Режиссер → Цена билета, Количество свободных мест – получается из приведенной выше ФЗ с помощью добавления атрибутов Название фильма и Режиссер.

Не являются ФЗ:

Название кинотеатра, Адрес, Номер зала → Дата, Время, Название фильма, Режиссер, Цена билета, Количество свободных мест – т.к. в одном зале проходит много сеансов.

Название кинотеатра, Адрес, Номер зала, Дата → Время, Название фильма, Режиссер, Цена билета, Количество свободных мест – т.к. в одном зале в один день может проходить несколько сеансов в разное время.

Название кинотеатра, Адрес, Номер зала, Время → Дата, Название фильма, Режиссер, Цена билета, Количество свободных мест – т.к. в одном зале в одно и то же время, но в разные дни могут показывать разные фильмы.

Название кинотеатра, Адрес, Номер зала, Название фильма, Режиссер → Дата, Время, Цена билета, Количество свободных мест – т.к. один и тот же фильм может показываться в зале в разные дни или время.

Отношение Сеанс находится в НФБК, т.к. все его детерминанты являются возможными ключами.

5) Роль (Фамилия, Имя, Дата рождения, Название фильма, Режиссер, Кого играет)

Функциональные зависимости:

Фамилия, Имя, Дата рождения, Название фильма, Режиссер → Кого играет – является ключом.

Не являются ФЗ:

Фамилия, Имя, Дата рождения, Название фильма → Режиссер, Кого играет – т.к. актер может сниматься в фильмах с одинаковыми названиями.

Имя, Фамилия, Дата рождения, Режиссер → Название фильма, Кого играет – т.к. актер может сниматься в нескольких фильмах одного режиссера.

Имя, Фамилия, Название фильма, Режиссер → Дата рождения– т.к. актеры с одним именем и фамилией могут сниматься в одном фильме.

Имя, Дата рождения, Название фильма, Режиссер → Фамилия – т.к. актеры с одним именем и датой рождения могут сниматься в одном фильме.

Фамилия, Дата рождения, Название фильма, Режиссер → Имя – т.к. актеры с одной фамилией и датой рождения могут сниматься в одном фильме.

Отношение Играет находится в НФБК, т.к. все его детерминанты являются возможными ключами.

6) Приз показываемого фильма (Название фильма, Режиссер, Название награды, Номинация, Год получения)

Функциональные зависимости:

Название фильма, Режиссер, Название награды, Номинация, Год получения → {}.

Не являются ФЗ:

Название фильма, Режиссер, Название награды, Номинация → Год получения – т.к. фильм может получить за разные годы призы в одной номинации.

Название фильма, Режиссер, Название награды, Год получения → Номинация – т.к. фильм может получить приз по нескольким номинациям в один год .

Название фильма, Режиссер, Номинация, Год получения → Название награды – т.к. фильм может выиграть несколько наград в одной номинации в один год .

Название фильма, Название награды, Номинация, Год получения → Режиссер – т.к. фильмы от разных режиссеров с одним названием могут получить одинаковые награды за один год.

Режиссер, Название награды, Номинация, Год получения → Название фильма – т.к. режиссер за разные фильмы за один год может выиграть несколько призов в одной номинации.

Отношение Приз показываемого фильма находится в НФБК, т.к. все его детерминанты являются возможными ключами.

7) Жанр фильма (Название фильма, Режиссер, Имя жанра)

Функциональные зависимости:

Название фильма, Режиссер, Имя жанра $\rightarrow \{ \}$

Не являются ФЗ:

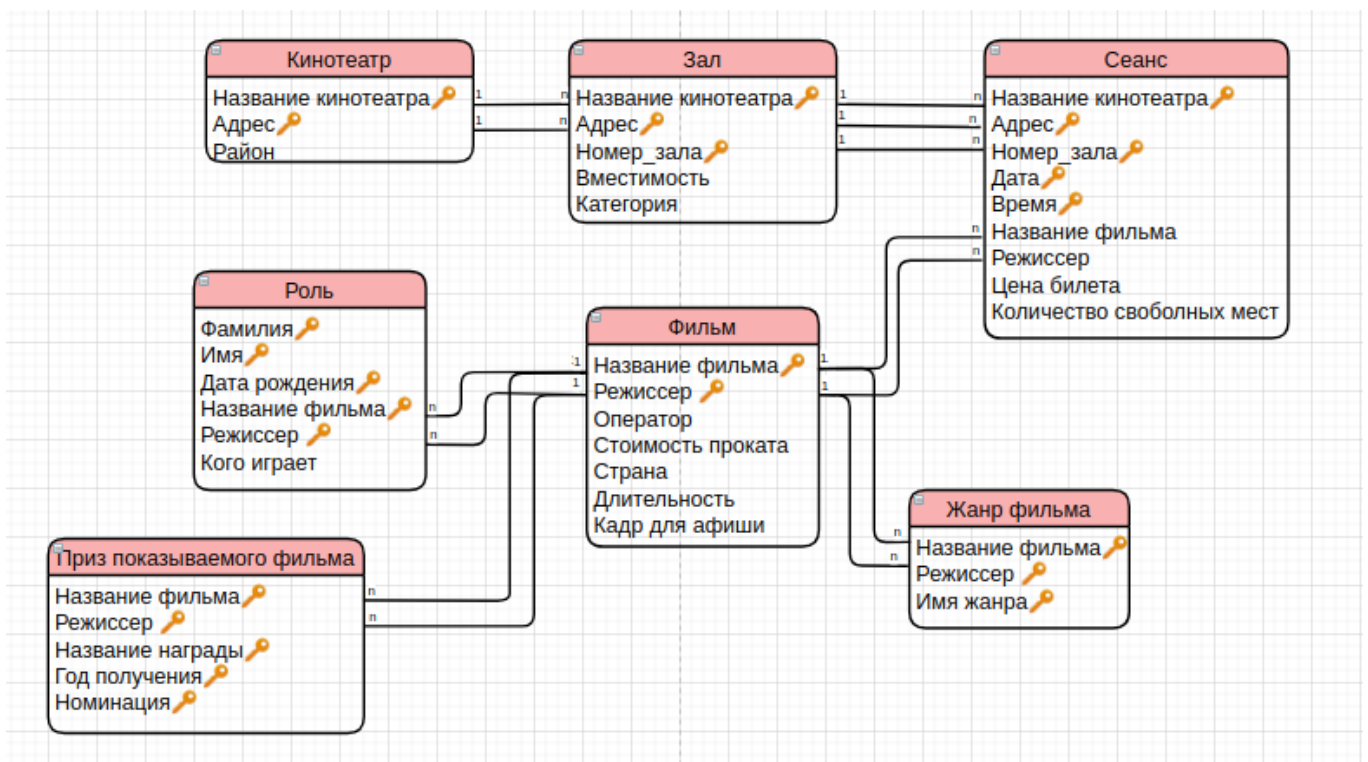
Название фильма, Режиссер \rightarrow Имя жанра – т.к. у фильма может быть несколько жанров.

Название фильма, Имя жанра \rightarrow Режиссер – т.к. одному жанру могут принадлежать фильмы с одним названием от разных режиссеров.

Режиссер, Имя жанра \rightarrow Название фильма – т.к. одному жанру могут принадлежать фильмы от одного режиссера с разными названиями.

Отношение Жанр фильма находится в НФБК, т.к. все детерминанты являются возможными ключами.

Схема:



3 Структуры таблиц и постоянные связи между таблицами

3.1 Структуры таблиц

Ниже представлены структуры всех таблиц спроектированной базы данных. Для упрощения структуры будем создавать дополнительные поля-идентификаторы типа счетчик. С кодом для создания таблиц можно ознакомиться в приложении А.

Таблица Кинотеатры (cinemas):

	cid	name	type	notnull	dflt_value	pk
1	0	id	INTEGER	1	<null>	1
2	1	name	TEXT(32)	1	<null>	0
3	2	address	TEXT(64)	1	<null>	0
4	3	district	TEXT(32)	1	<null>	0
5	4	is_open	INTEGER	1	1	0

Таблица Залы (rooms):

	cid	name	type	notnull	dflt_value	pk
1	0	id	INTEGER	1	<null>	1
2	1	cinema_id	INTEGER	1	<null>	0
3	2	number	INTEGER	1	<null>	0
4	3	capacity	INTEGER	1	<null>	0
5	4	category	INTEGER	1	<null>	0

Таблица Фильмы (films):

	cid	name	type	notnull	dflt_value	pk
1	0	id	INTEGER	1	<null>	1
2	1	name	TEXT(32)	1	<null>	0
3	2	producer	TEXT(32)	1	<null>	0
4	3	operator	TEXT(32)	1	<null>	0
5	4	cost	INTEGER	1	<null>	0
6	5	country	TEXT(32)	1	<null>	0
7	6	duration	INTEGER	1	<null>	0
8	7	picture	TEXT(512)	1	<null>	0

Таблица Роли (roles):

	cid	name	type	notnull	dflt_value	pk
1	0	id	INTEGER	1	<null>	1
2	1	name	TEXT(32)	1	<null>	0
3	2	surname	TEXT(32)	1	<null>	0
4	3	birth	TEXT(19)	1	<null>	0
5	4	film_id	INTEGER	1	<null>	0
6	5	role_name	TEXT(32)	1	<null>	0

Таблица Сеансы (sessions):

	cid	name	type	notnull	dflt_value	pk
1	0	id	INTEGER	1	<null>	1
2	1	room_id	INTEGER	1	<null>	0
3	2	film_id	INTEGER	1	<null>	0
4	3	date_time	TEXT(19)	1	<null>	0
5	4	seats	INTEGER	1	<null>	0

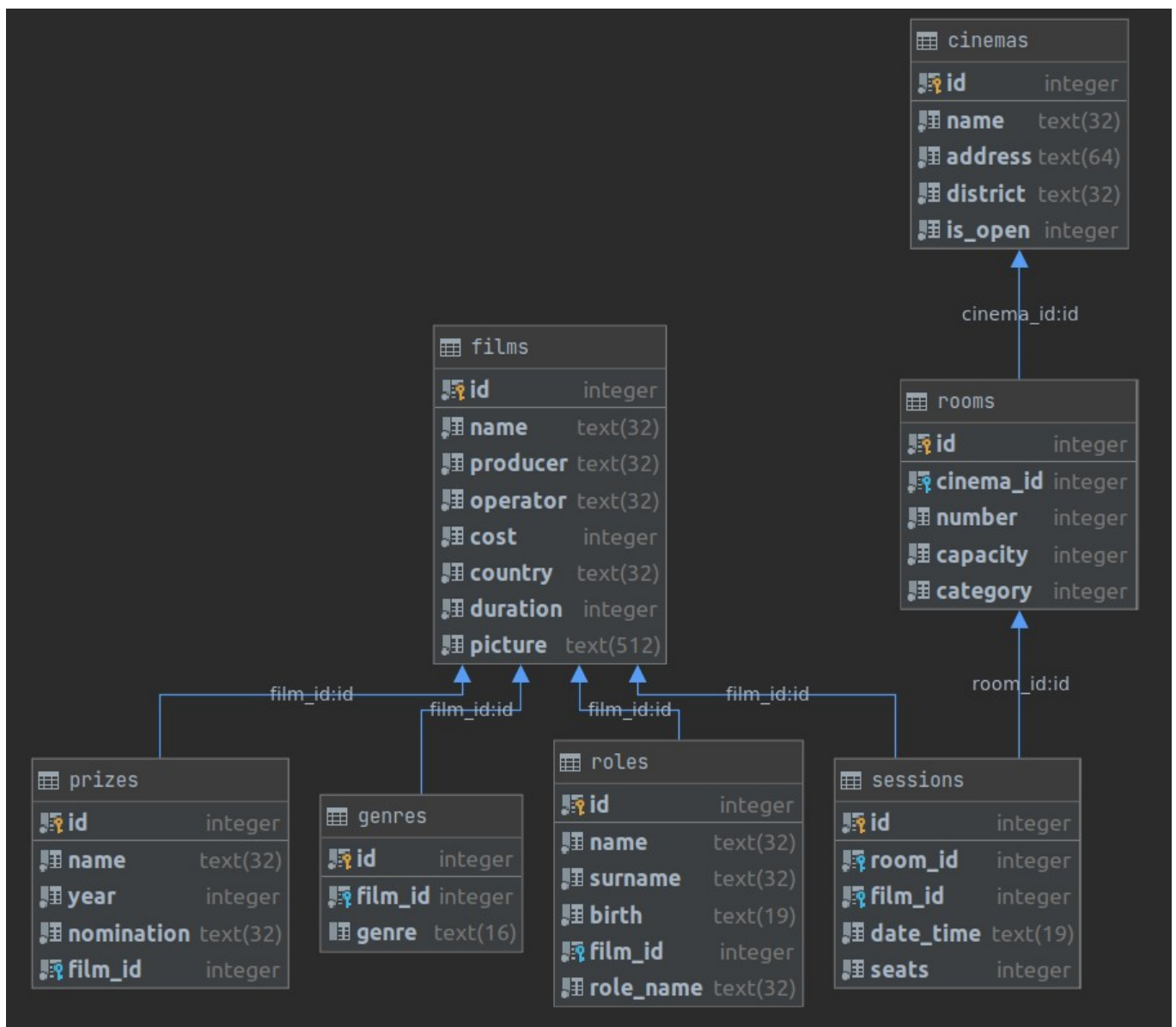
Таблица Призы (prizes):

	cid	name	type	notnull	dflt_value	pk
1	0	id	INTEGER	1	<null>	1
2	1	name	TEXT(32)	1	<null>	0
3	2	year	INTEGER	1	<null>	0
4	3	nomination	TEXT(32)	1	<null>	0
5	4	film_id	INTEGER	1	<null>	0

Таблица Жанры (genres):

	cid	name	type	notnull	dflt_value	pk
1	0	id	INTEGER	1	<null>	1
2	1	film_id	INTEGER	1	<null>	0
3	2	genre	TEXT(16)	0	<null>	0

3.2 Постоянные связи между таблицами



4 Содержимое таблиц

С кодом, с помощью которого заполнялись таблицы можно ознакомиться в приложении Б.

Cinemas:

	id	name	address	district	is_open
1	1	мир	победы 1	центральный	1
2	2	победа	пушкина 8	приморский	1
3	3	премьера	ленина 88	центральный	1
4	4	эльдар	ленина 256	приморский	1
5	5	искра	пушкина 44	центральный	1

Films:

	id	name	producer	operator	cost	country	duration	picture
1	1	На игле	Денни Бойл	Брайан Тьюфани	3000000	Великобритания	93	https://avatars.mds.yandex.net/get-kinopoisk-image/177776...
2	2	Страх и ненависть в Лас-Вегасе	Терри Гиллиам	Никола Лекорини	5000000	США	110	https://b1.filmpro.ru/c/8866.jpg
3	3	Джентельмены	Гай Ричи	Алан Стюарт	2000000	Великобритания	113	https://avatars.mds.yandex.net/get-kinopoisk-image/159902...
4	4	Карты, деньги, два ствола	Гай Ричи	Тим Морис-Джонс	1000000	Великобритания	115	https://b1.filmpro.ru/c/458171.jpg
5	5	Криминальное чтиво	Квентин Тарантино	Анджей Секула	2500000	США	154	https://upload.wikimedia.org/wikipedia/ru/9/93/Pulp_Ficti...
6	6	Интерстеллар	Кристофер Нолан	Хойте Ван Хойтема	6000000	США	169	https://avatars.mds.yandex.net/get-kinopoisk-image/160064...
7	7	Золото	Стивен Гейган	Роберт Элсвит	4000000	США	120	https://avatars.mds.yandex.net/get-kinopoisk-image/190078...

Rooms:

ms	id	cinema_id	number	capacity	category
1	1	1	1	40	1
2	2	1	2	80	2
3	3	1	3	30	1
4	4	2	1	50	3
5	5	2	2	60	3
6	6	3	1	40	1
7	7	3	2	50	3
8	8	4	1	20	2
9	9	4	2	40	1
10	10	4	3	20	2
11	11	4	4	30	2
12	12	5	1	40	1
13	13	5	2	50	1
14	14	5	3	60	2
15	15	5	4	70	3

Sessions:

	🎫 id ↕	🎫 room_id ↕	🎫 film_id ↕	📅 date_time ↕	🎫 seats ↕
1	1	1	1	2021-01-10 12:00:00	10
2	2	2	1	2021-01-10 12:00:00	1
3	3	3	2	2021-01-10 12:00:00	5
4	4	4	2	2021-01-10 13:00:00	4
5	5	5	3	2021-01-10 13:00:00	5
6	6	6	3	2021-01-11 13:00:00	7
7	7	7	4	2021-01-11 14:00:00	3
8	8	8	5	2021-01-11 14:00:00	4
9	9	9	1	2021-01-11 14:00:00	5
10	10	10	1	2021-01-11 15:00:00	1
11	11	11	3	2021-01-10 15:00:00	3
12	12	12	5	2021-01-10 15:00:00	10
13	13	13	2	2021-01-10 16:00:00	5
14	14	14	4	2021-01-10 16:00:00	6
15	15	15	2	2021-01-10 16:00:00	6
16	16	1	5	2021-01-11 17:00:00	5
17	17	2	5	2021-01-11 17:00:00	8
18	18	3	5	2021-01-11 17:00:00	18
19	19	5	5	2021-01-11 18:00:00	9
20	20	6	5	2021-01-10 18:00:00	10
21	21	7	5	2021-01-10 19:00:00	0
22	22	8	1	2021-01-10 19:00:00	0
23	23	9	3	2021-01-10 19:00:00	0
24	24	10	2	2021-01-10 20:00:00	4
25	25	11	2	2021-01-10 20:00:00	5
26	26	12	2	2021-01-10 20:00:00	1
27	27	13	2	2021-01-10 21:00:00	5
28	28	14	4	2021-01-10 21:00:00	6
29	29	15	4	2021-01-10 21:00:00	3

Roles:

	id	name	surname	birth	film_id	name
1	1	Юэн	Макгрегор	1989-10-01		Пин Таб
2	2	Роберт	Карлайл	1972-01-10	1	Френсис Бегби
3	3	Джонни	Депп	1975-04-15	2	Рауль Дюк
4	4	Бенисио	дель Торо	1969-11-05	2	Доктор Гонзо
5	5	Чарли	Ханнем	1980-12-21	3	Реймонд Смит
6	7	Колин	Фарелл	1969-10-29	3	Тренер
7	8	Джейсон	Стейтем	1962-10-07	4	Бейкон
8	9	Джон	Траволта	1956-11-05	5	Винсент Вега
9	10	Ума	Турман	1986-10-15	5	Мия Уоллес
10	11	Сэмюэл	Лерой Джексон	1946-12-27	5	Джулс Уиннфилд
11	12	Брюс	Уиллис	1966-01-02	5	Бутч Куллидж
12	13	Тим	Рот	1980-01-20	5	Тыковка
13	15	Энн	Хэтүэй	1985-10-24	6	Амелия Бренд

Prizes:

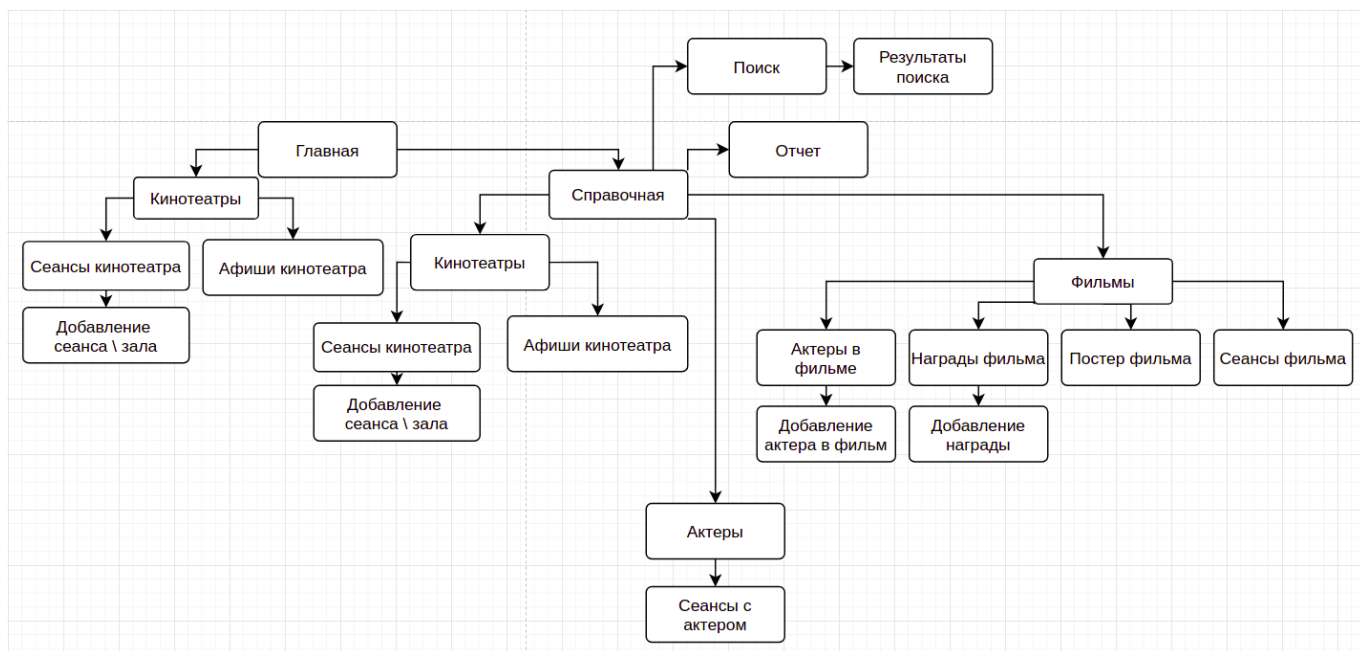
	id	name	year	nomination	film_id
1	1	Премия BAFTA	1996	Лучший адаптированный сценарий	1
2	2	Золотая пальмовая ветвь	1998		2
3	3	Премия BAFTA	1999	Выдающийся британский фильм года	4
4	4	Премия британского независимого кино	1998	Лучший британский независимый фильм	4
5	5	Золотая пальмовая ветвь	1994		5
6	6	Оскар	1995	лучший оригинальный сценарий	5
7	7	Оскар	2015	лучшие визуальные эффекты	6
8	8	Империя	2015	лучший фильм	6

Genres:

	id	film_id	genre
1	26	1	Комедия
2	27	1	Драма
3	28	1	Криминал
4	29	2	Комедия
5	30	2	Сатира
6	31	2	Приключения
7	32	2	Драма
8	33	3	Боевик
9	34	3	Комедия
10	35	3	Криминал
11	36	4	Комедия
12	37	4	Триллер
13	38	4	Криминал
14	39	5	Комедия
15	40	5	Криминал
16	41	5	Триллер
17	42	5	Драма
18	43	6	Фантастика
19	44	6	Приключения
20	45	6	Драма
21	46	6	Детектив
22	47	7	Приключения
23	48	7	Триллер
24	49	7	Драма
25	50	7	Криминал

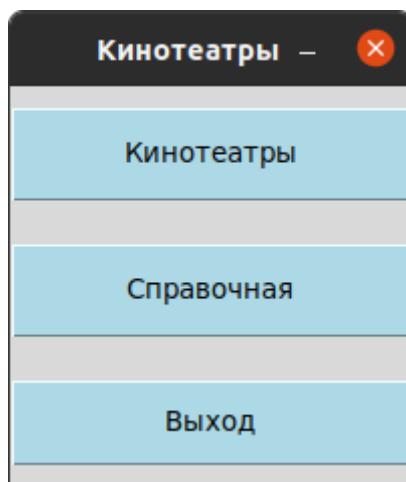
5 Схема иерархии интерфейса

Для реализации графического пользовательского интерфейса был избран язык Python3.8 и библиотека tkinter для работы с БД используется библиотека SQLite.



6 Описание окон

6.1 Главная (Root)



Назначение: предоставляет возможность выбрать пользователю роль или выйти из приложения.

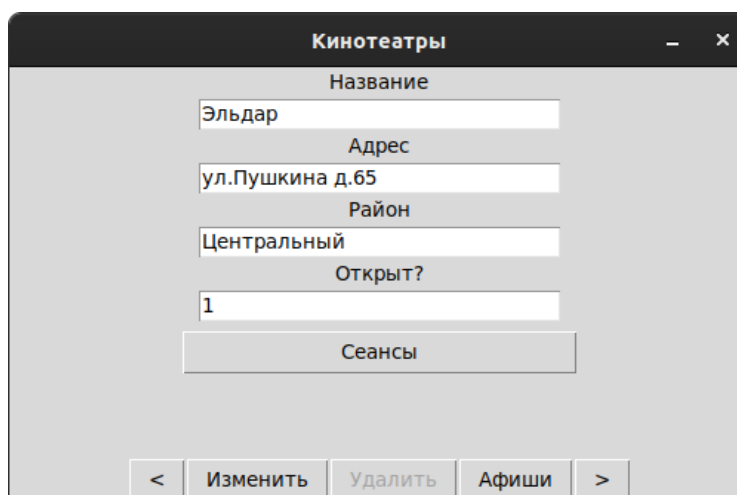
Кнопки:

Кинотеатр – по нажатию создается экземпляр класса CinemaWindow.

Справочная – по нажатию создается экземпляр класса CinemaWindow с флагом подтверждающим, что это действительно справочная служба.

Выход – по нажатию происходит выход из приложения.

6.2 Кинотеатр – Кинотеатры (CinemaWindow)



Назначение: Перечень кинотеатров с возможностью изменения, но без возможности удаления и добавления кинотеатров. В данном окне можно ознакомиться с адресом и районом кинотеатра, а также с его репертуаром если нажать на кнопку Сеансы.

Кнопки:

Сеансы – по нажатию создается экземпляр класса CinemaSessions.

Изменить – по нажатию происходит изменение соответствующего кинотеатра в базе данных.

Афиши – по нажатию появляется окно позволяющее ознакомиться с подробнейшим описанием фильма, и временем сеансов когда данный фильм будет показываться.

Источник данных:

```
SELECT * FROM {table}
```

6.3 Кинотеатр – Сеансы (CinemaSessions)

Эльдар7					
№	Фильм	Дата и время сеанса	Цена	Места	
1	На игле	2021-01-10 15:00:00	300	10	DEL
2	Страх и ненависть в Л	2021-01-10 16:00:00	200	20	DEL
3	Дежнтельмены	2021-01-10 16:00:00	600	5	DEL
3	Дежнтельмены	2021-01-13 18:00:00	600	2	DEL
2	Криминальное чтиво	2021-01-14 12:00:00	400	5	DEL
1	Страх и ненависть в Л	2021-01-14 15:00:00	300	6	DEL
			Обновить		
			Добавить		

Назначение: Демонстрация репертуара кинотеатра с информацией о наличии свободных мест на данный сеанс.

Кнопки:

DEL – по нажатию происходит удаление из базы данных соответствующей записи.

Обновить – по нажатию происходит обновление базы данных в соответствии с изменениями.

Добавить – по нажатию создается экземпляр класса CinemaInsertSession.

Источник данных:

```
SELECT sessions.id, rooms.number, name, date_time, films.cost /
10000 * (4-rooms.category) AS cost, seats
FROM sessions INNER JOIN films ON film_id = films.id
INNER JOIN rooms ON room_id = rooms.id
WHERE room_id IN (SELECT id
FROM rooms
WHERE cinema_id="{cinema_id}" OR
cinema_id={cinema_id});
```

6.4 Кинотеатр – Афиши (PostersWindow)

Афиши

Криминальное чтиво (США)

Режиссер: Квентин Тарантино

Оператор: Анджей Секула

Продолжительность: 154 мин.


Жанры: Комедия
Криминал
Триллер

Призы

- * Оскар 1995
Лучший сценарий
- * Золотой глобус 1995
Лучший сценарий
- * Каннский кинофестиваль 1994
Золотая пальмовая ветвь

Сеансы

Начало сеанса	Свободные места	Цена билета
2021-01-14 12:00:00	1	400



В ролях

Джон Траволта – Винсент Вега
Сэмюэл Джексон – Джулс Уиннфилд

<

>

Назначение: окно демонстрирующее афиши для всех фильмов заданного кинотеатра.

Источники данных:

Для получения информации о фильме в целом:

```
SELECT films.id, films.name, films.producer, films.operator,  
films.country, films.duration, GROUP_CONCAT(DISTINCT genres.genre),  
films.picture  
FROM films INNER JOIN sessions ON sessions.film_id = films.id  
INNER JOIN rooms ON rooms.id = sessions.room_id AND  
rooms.cinema_id="{cinema_id}"  
INNER JOIN genres ON films.id = genres.film_id  
GROUP BY films.id;
```

Для получения информации о призах:

```
SELECT * FROM prizes WHERE prizes.film_id="{str(film_id)}";
```

Для получения информации об актерах:

```
SELECT name, surname, role_name  
FROM roles  
WHERE film_id = "{str(film_id)}";
```

Для получения информации о сеансах:

```
SELECT sessions.id, cinemas.name, rooms.number, date_time, seats,  
films.cost / 10000 * (4-rooms.category) AS cost  
FROM sessions INNER JOIN films ON film_id = films.id  
INNER JOIN rooms ON room_id = rooms.id  
INNER JOIN cinemas ON rooms.cinema_id = cinemas.id  
WHERE (film_id = "{film_id}" OR film_id = {film_id}) AND  
(cinemas.id = "{cinema_id}" OR cinemas.id = {cinema_id});
```

6.5 Кинотеатр – Добавление зала или сеанса (CinemaInsertSession)

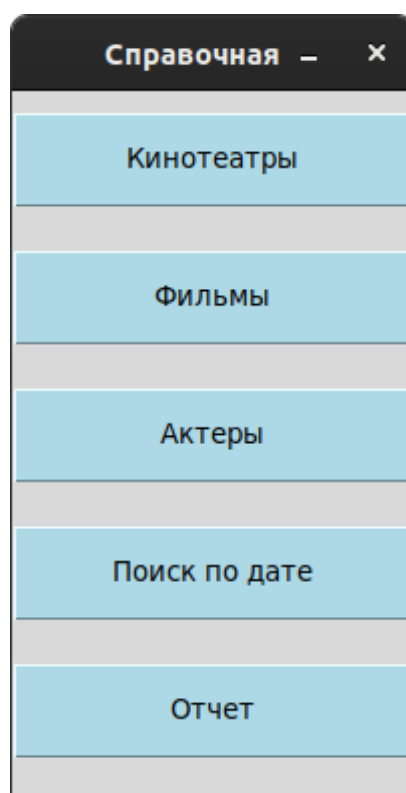
The screenshot shows a window titled "Эльдар7" with standard Windows window controls. Inside the window, there are two main sections: "Зал" (Hall) and "Сеанс" (Session). The "Зал" section contains three input fields labeled "Номер" (Number), "Вместимость" (Capacity), and "Категория" (Category), followed by a "Добавить зал" (Add hall) button. The "Сеанс" section contains five input fields labeled "Номер зала" (Hall number), "Название фильма" (Movie title), "Режиссер" (Director), "Дата и время сеанса" (Date and time of session), and "Места" (Seats), followed by a "Добавить сеанс" (Add session) button.

Назначение: окно позволяющее добавлять как залы для кинотеатров, так и сеансы.

Кнопки:

Добавить зал / Добавить сеанс – по нажатию происходит добавление в базу данных соответствующей записи.

6.6 Справочная служба – Главное окно (FaqWindow)



Назначение: Главное окно справочной службы.

Кнопки:

Кинотеатры – по нажатию создается экземпляр класса CinemaWindow с флагом, который вызывает активацию кнопок Добавить и Удалить.

Фильмы – по нажатию создается экземпляр класса FilmsWindow, в создающемся окне можно ознакомиться с фильмами находящимися в прокате.

Актеры – по нажатию создается экземпляр класса ActorsWindow, в создающемся окне можно ознакомиться с актерами, которые поучаствовали в показываемых фильмах.

Поиск по дате – по нажатию создается экземпляр класса SearchWindow, с помощью этого окна можно найти фильмы, которые можно посмотреть в заданный день в определенное время.

Отчет – по нажатию создается экземпляр класса ReportWindow, в окне будет выведен отчет о доходности кинотеатров.

6.7 Справочная служба – Кинотеатры (CinemaWindow)

The screenshot shows a window titled "Кинотеатры" (Cinemas) with a dark header bar containing a minus sign and a close button. The main area has a light gray background. It contains a form with the following elements:

- Label: "Название" (Name)
- Text input: "Эльдар" (El'dar)
- Label: "Адрес" (Address)
- Text input: "ул.Пушкина д.65" (Pushkin St. 65)
- Label: "Район" (District)
- Text input: "Центральный" (Central)
- Label: "Открыт?" (Open?)
- Text input: "1"
- Text button: "Сеансы" (Sessions)

At the bottom, there is a row of five buttons: "<", "Изменить" (Edit), "Удалить" (Delete), "Афиши" (Posters), and ">".

Последняя страница выглядит следующим образом:

The screenshot shows the same "Кинотеатры" window, but with a different set of buttons at the bottom. The form fields are identical to the previous screenshot:

- Label: "Название" (Name)
- Text input: (empty)
- Label: "Адрес" (Address)
- Text input: (empty)
- Label: "Район" (District)
- Text input: (empty)
- Label: "Открыт?" (Open?)
- Text input: (empty)
- Text button: "Сеансы" (Sessions)

At the bottom, the buttons are: "<", "Добавить" (Add), "Удалить" (Delete), "Афиши" (Posters), and ">".

Назначение: В данном окне можно ознакомиться с адресом и районом кинотеатра, а также с его репертуаром и афишами.

Кнопки:

Сеансы – по нажатию создается экземпляр класса CinemaSessions.

Изменить – по нажатию происходит изменение соответствующего кинотеатра в базе данных.

Удалить – по нажатию происходит удаление соответствующего кинотеатра в базе данных.

Добавить – по нажатию происходит добавление кинотеатра в базу данных в соответствие с введенными данными.

Источники данных такие же как и в случае открытия окна через кинотеатры.

6.8 Справочная служба – Сеансы (CinemaSessions)

Окно ничем не отличающееся от окна в п. 6.3

6.9 Справочная служба – Афиши (PostersWindow)

Окно ничем не отличающееся от окна в п. 6.4

6.10 Справочная служба – Добавление зала или сеанса (CinemaInsertSession)

Окно ничем не отличающееся от окна в п. 6.5

6.11 Справочная служба – Перечень фильмов (FilmsWindow)

The screenshot shows a window titled "Фильмы" (Films) with a dark header bar containing a minus sign and a close button. The main area is light gray and contains several text input fields, each with a label above it. The fields are filled with the following data:

Название	Режиссер	Оператор	Стоимость	Страна	Продолжительность	Ссылка на кадр	Жанры
На игле	Денни Бойл	Брайан Тьюфано	1000000	Великобритания	93	https://thumbs.dfs.ivi.ru/storage33/contents/4/a/6	Криминал, Комедия, Драма

Below the fields are four buttons: "Актеры", "Призы", "Показать кадр", and "Сеансы". At the bottom of the window are four buttons: "<", "Изменить", "Удалить", and ">".

Назначение: В данном окне можно ознакомиться с названием, режиссером, оператором, прокатной стоимостью, происхождением, продолжительностью, а также ссылкой на кадр для афиши.

Кнопки:

Актеры – по нажатию открывается окно позволяющее ознакомиться с участвующими актерами и их ролями в этом фильме.

Призы – по нажатию открывается окно позволяющее ознакомиться с наградами фильма.

Показать кадр – по нажатию открывается окно позволяющее увидеть постер фильма.

Сеансы – по нажатию открывается окно позволяющее ознакомиться со всеми сеансами фильма во всех кинотеатрах.

Удалить – по нажатию происходит удаление соответствующего фильма в базе данных.

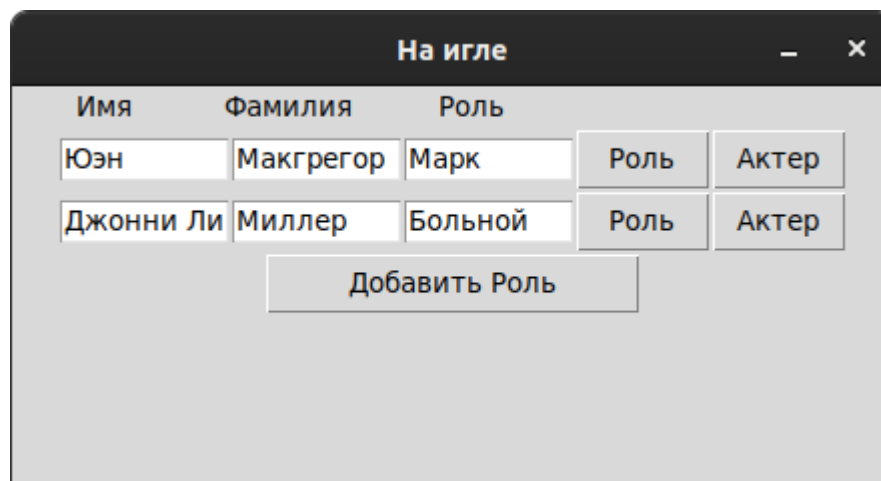
Изменить – по нажатию происходит изменение фильма в базе данных в соответствие с введенными данными.

Источники данных:

```
SELECT * FROM films;
```

```
SELECT genre FROM genres WHERE film_id="{str(film_id)}"
```

6.12 Справочная служба – Роли фильма (FilmsActors)



Имя	Фамилия	Роль
Юэн	Макгрегор	Марк
Джонни Ли	Миллер	Больной

Добавить Роль

Назначение: демонстрация актеров и их ролей в фильме.

Кнопки:

Роль/Актер – при нажатии удаляется либо запись о роли либо об актере в целом.

Добавить роль – при нажатии создается окно позволяющее добавить роль к заданному фильму.

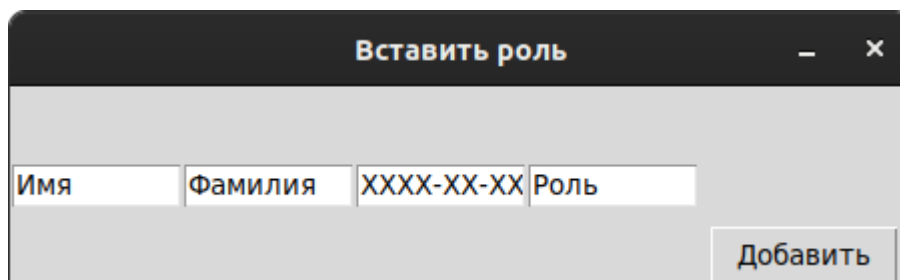
Источники данных:

```
SELECT name, surname, role_name
```

```
FROM roles
```

```
WHERE film_id = "{str(film_id)}";
```

6.13 Справочная служба – Добавление роли (FilmsInsertRole)

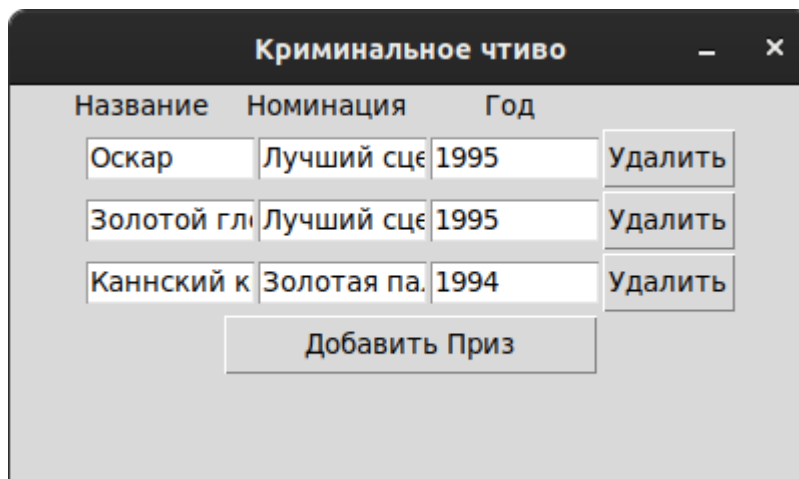


Назначение: Добавление роли.

Кнопки:

Добавить – по нажатию происходит добавление актера (если нет записей о нем) и роли в БД.

6.14 Справочная служба – Призы фильма (FilmsPrizes)



Название	Номинация	Год	
Оскар	Лучший сце	1995	Удалить
Золотой гло	Лучший сце	1995	Удалить
Каннский к	Золотая па.	1994	Удалить

Назначение: демонстрация призов в фильма.

Кнопки:

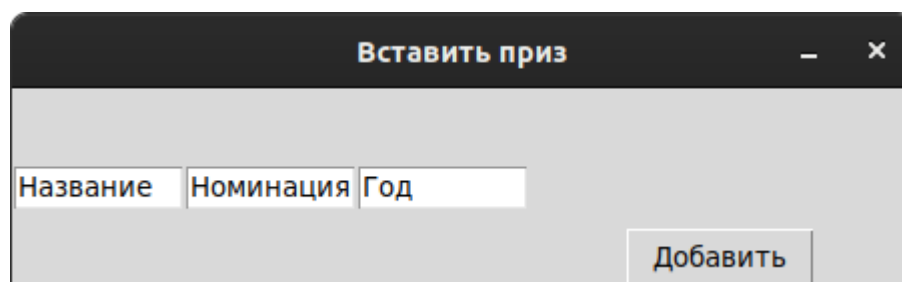
Удалить – по нажатию происходит удаление соответствующего приза.

Добавить приз – по нажатию открывается окно позволяющее ввести данные о награде данного фильма и затем добавить их в БД.

Источники данных:

```
SELECT * FROM prizes WHERE prizes.film_id="{str(film_id)}"
```

6.15 Справочная служба – Добавление приза (FilmsInsertPrize)



Вставить приз

Название Номинация Год

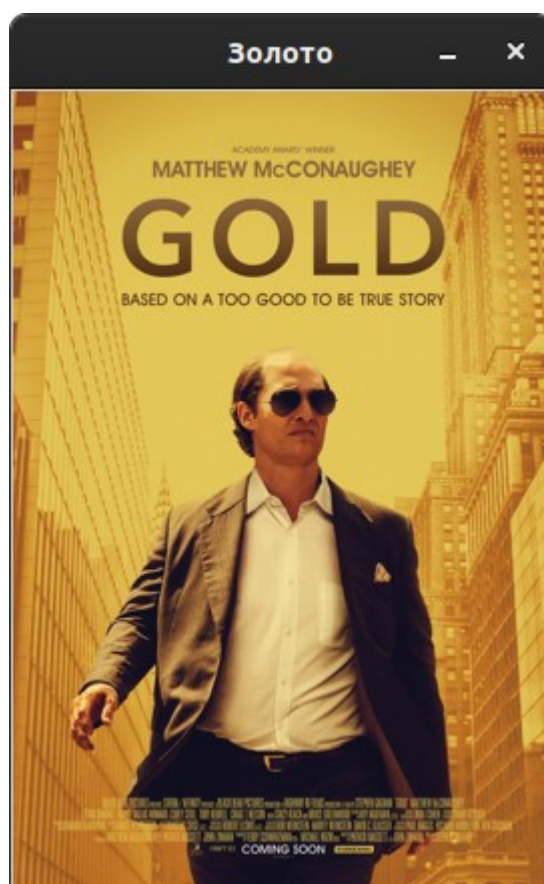
Добавить

Назначение: Добавление награды.

Кнопки:

Добавить – по нажатию происходит добавление приза в БД.

6.16 Справочная служба – Постер фильма (FilmPoster)



6.17 Справочная служба – Сеансы фильма

На игле				
Кинотеатр	Зал	Дата и время	Места	Цена
Мир	1	2021-01-10 19:00:00	5	200
Эльдар	1	2021-01-12 17:00:00	4	300

Назначение: Демонстрация сеансов, на которых будет показываться данный фильм, с информацией о наличии свободных мест на данный сеанс и ценой.

Источник данных:

```
SELECT sessions.id, cinemas.name, rooms.number, date_time, seats,
films.cost / 10000 * (4-rooms.category) AS cost
FROM sessions INNER JOIN films ON film_id = films.id
INNER JOIN rooms ON room_id = rooms.id
INNER JOIN cinemas ON rooms.cinema_id = cinemas.id
WHERE film_id = "{film_id}" OR film_id = {film_id};
```

6.18 Справочная служба – Перечень актеров

Актеры

Имя

Джейсон

Фамилия

Стейтем

Дата рождения

1967-07-26

Сеансы

<

Изменить

Удалить

>

Назначение: Демонстрация актеров поучаствовавших в производстве показываемых фильмов.

Кнопки:

Сеансы – по нажатию создается экземпляр класса ActorSessions.

Изменить – по нажатию происходит изменение соответствующего актера в базе данных.

Удалить – по нажатию происходит удаление соответствующего актера в базе данных.

Источник данных:

```
SELECT DISTINCT name, surname, birth FROM roles;
```

6.19 Справочная служба – Сеансы при участии заданного актера

Сеансы п.у. Мэттью Макконахи					
Название фильма	Кинотеатр	Зал	Дата и время	Места	Цена
Дежнтельмены	Эльдар	3	2021-01-13 18:00:00	6	600
Дежнтельмены	Эльдар	1	2021-10-11 18:00:00	12	600
Дежнтельмены	Эльдар	3	2021-01-10 16:00:00	3	600
Дежнтельмены	Премьера	2	2021-01-12 15:00:00	9	200
Золото	Эльдар	1	2000-12-13 18:00:00	3	3000
Интерстеллар	Победа	1	2020-05-09 09:00:00	10	200

Назначение: демонстрация сеансов, на которых можно посмотреть фильмы с участием заданного актера.

Источник данных:

```
SELECT sessions.id, films.name, cinemas.name, rooms.number,  
date_time, seats, films.cost / 10000 * (4-rooms.category) AS cost  
FROM sessions INNER JOIN films ON sessions.film_id = films.id  
INNER JOIN roles ON roles.film_id = films.id  
INNER JOIN rooms ON room_id = rooms.id  
INNER JOIN cinemas ON rooms.cinema_id = cinemas.id  
WHERE roles.name = "{actor[0]}" AND roles.surname = "{actor[1]}"  
AND roles.birth = "{actor[2]}";
```

6.20 Справочная служба – Поиск

Поиск

Дата

XXXX-XX-XX

С

XX:XX:XX

До

XX:XX:XX

Жанр

Поиск

Поиск

Дата

2021-01-10

С

00:00:00

До

23:00:00

Жанр

Комедия

Поиск

Результаты поиска						
Кинотеатр	Название фильма	Зал	Дата и время	Места	Цена	Жанры
Страх и ненависть	Эльдар	2	2021-01-10 16:00:00	2	200	Драма, Комедия
Дежнтельмены	Эльдар	3	2021-01-10 16:00:00	3	600	Боевик, Комедия, I
Карты, деньги, д	Каро	1	2021-01-10 17:00:00	11	60	Боевик, Комедия, I
Криминальное чт	Каро	2	2021-01-10 18:00:00	10	200	Комедия, Кримин
На игле	Мир	1	2021-01-10 19:00:00	5	200	Драма, Комедия, К
Страх и ненависть	Премьера	1	2021-01-10 20:00:00	8	300	Драма, Комедия

Назначение: поиск сеансов в заданный день в определенный промежуток времени с фильтрацией фильмов по жанру.

Источник данных:

```
SELECT sessions.id, f.name, c.name, r.number, date_time, seats,
f.cost / 10000 * (4-r.category) AS cost, GROUP_CONCAT(g.genre)
FROM sessions INNER JOIN rooms r ON r.id = sessions.room_id
INNER JOIN cinemas c ON c.id = r.cinema_id
INNER JOIN films f ON f.id = sessions.film_id AND "{genre}" IN
(SELECT genre FROM genres WHERE f.id = genres.film_id)
INNER JOIN genres g ON f.id = g.film_id
GROUP BY sessions.id
```

```
HAVING date_time > "{date} {time_b}" AND date_time < "{date} {time_e}"
```

Кнопки:

Поиск – запускает поиск сеансов и выводит найденное в окне.

6.21 Справочная служба – Отчет

Отчет					-	□	×
Отчет по районам							
Суммарный доход: -26868700.0							
Приморский							
Доход: -2584660.0							
Каро							
Доход: -2584660.0							
Карты, деньги, два ствола				Кол-во сеансов:2	Средняя цена:60.0		
Криминальное чтиво				Кол-во сеансов:1	Средняя цена:200.0		
Фрунзенский							
Доход: -5553940.0							
Премьера							
Доход: -5553940.0							
Дежтельмены				Кол-во сеансов:1	Средняя цена:200.0		
Карты, деньги, два ствола				Кол-во сеансов:1	Средняя цена:180.0		
Криминальное чтиво				Кол-во сеансов:1	Средняя цена:200.0		
Страх и ненависть в Лас-Вегасе				Кол-во сеансов:1	Средняя цена:300.0		
Центральный							
Доход: -18730100.0							
Мир							
Доход: -993000.0							
На игле				Кол-во сеансов:1	Средняя цена:200.0		
Победа							
Доход: -1982000.0							
Интерстеллар				Кол-во сеансов:1	Средняя цена:200.0		
Эльдар							
Доход: -15755100.0							
Дежтельмены				Кол-во сеансов:3	Средняя цена:600.0		
Золото				Кол-во сеансов:1	Средняя цена:3000.0		
Криминальное чтиво				Кол-во сеансов:1	Средняя цена:400.0		
На игле				Кол-во сеансов:1	Средняя цена:300.0		
Страх и ненависть в Лас-Вегасе				Кол-во сеансов:3	Средняя цена:233.33		
Отчет по жанрам							
Боевик		2					
Драма		3					
Комедия		5					
Криминал		4					
Приключения		2					
Триллер		2					
Фантастика		1					

Назначение: вывод отчета, в котром демонстрируется доход по всему городу, по районам и по отдельным кинотеатрам. Также для каждого фильма выводится количество сеансов и средняя цена за билет. В конце предоставляется отчет содержащий жанры и количество фильмов под них подходящих.

Источники данных:

Для получения информации о фильмах по кинотеатру:

```
SELECT c.district, c.name, f.name, avg((4 - r.category) *
f.cost/10000) AS avg_cost, count(s.id)
FROM films INNER JOIN sessions s ON films.id = s.film_id
INNER JOIN rooms r ON s.room_id = r.id
INNER JOIN cinemas c ON c.id = r.cinema_id
INNER JOIN films f ON f.id = s.film_id
```

```
GROUP BY cinema_id, film_id
ORDER BY district, c.name, f.name;
```

Для получения информации о доходе каждого кинотеатра:

```
SELECT sum(income), cinema, district
FROM (SELECT c.name as cinema, c.district as district,
sum(r.capacity - sessions.seats) * avg((4 - r.category) * f.cost
/10000) - f.cost as income
      FROM sessions INNER JOIN rooms r ON r.id = sessions.room_id
      INNER JOIN films f ON f.id = sessions.film_id
      INNER JOIN cinemas c ON c.id = r.cinema_id
      GROUP BY film_id, r.cinema_id
      ORDER BY r.cinema_id)
GROUP BY cinema
ORDER BY district, cinema;
```

Для получения информации о доходе каждого района:

```
SELECT district, sum(income)
FROM (SELECT sum(income) AS income, district
      FROM (SELECT c.name AS cinema, c.district AS district,
sum(r.capacity-sessions.seats) * avg(f.cost/10000 * (4 - r.category))
- f.cost AS income
      FROM sessions INNER JOIN rooms r ON r.id = sessions.room_id
      INNER JOIN films f ON f.id = sessions.film_id
      INNER JOIN cinemas c ON c.id = r.cinema_id
      GROUP BY film_id, r.cinema_id
      ORDER BY r.cinema_id)
      GROUP BY cinema)
GROUP BY district;
```

Для получения информации о доходе всего города:

```
SELECT SUM(income)
FROM (SELECT district, sum(income) AS income
      FROM (SELECT sum(income) AS income, district
      FROM (SELECT c.name AS cinema, c.district AS district,
sum(r.capacity-sessions.seats) * avg(f.cost/10000 * (4 - r.category))
- f.cost AS income
      FROM sessions INNER JOIN rooms r ON r.id =
```

```
sessions.room_id
    INNER JOIN films f ON f.id = sessions.film_id
    INNER JOIN cinemas c ON c.id = r.cinema_id
    GROUP BY film_id, r.cinema_id
    ORDER BY r.cinema_id)
GROUP BY cinema)
GROUP BY district);
```

Для получения информации о жанрах:

```
SELECT genre, count(film_id) FROM genres GROUP BY genre;
```

7 Описание запросов

Код используемый для работы с базой данных представлен в приложении В. Все методы можно разбить на две группы – те, что дают некую информацию и те, что изменяют записи в таблице или удаляют или вставляют записи. Для каждой группы был реализован свой декоратор, который расширял эти методы так, чтобы корректно работать с базой данных. Запросы первой группы были представлены в предыдущем разделе, в этом разделе будут рассмотрены запросы вызывающие изменение базы данных.

7.1 Обновление кинотеатра

```
UPDATE cinemas
SET name="{name}", address="{address}", district="{district}",
is_open="{is_open}"
WHERE id="{cinema_id}";
```

Запрос изменяет название, адрес, район и статус кинотеатра в соответствии с полученным идентификатором.

7.2 Обновление сеанса

```
UPDATE sessions
SET room_id="{room_id}", date_time="{date_time}", seats="{seats}"
WHERE id="{session_id}"
```

Запрос изменяет идентификатор зала, дату и время сеанса, количество мест в соответствии с полученным идентификатором.

7.3 Обновление фильма

```
UPDATE films
SET name="{name}", producer="{prod}", operator="{operator}",
cost="{str(cost)}", country="{country}", duration="{str(duration)}",
picture="{pic}"
WHERE id="{str(film_id)}"
```

Запрос изменяет название, режиссера, оператора, стоимость, страну производства, продолжительность и кадр фильма в соответствии с полученным идентификатором.

7.4 Обновление актера

```
UPDATE roles
SET name="{name}", surname="{surname}", birth="{birth}"
WHERE name="{old[0]}" AND surname="{old[1]}" AND birth="{old[2]}"
```

Запрос изменяет имя, фамилию и дату рождения актера в соответствии с полученными старыми значениями.

7.5 Вставка кинотеатра

```
INSERT INTO cinemas (name, address, district, is_open)
VALUES ("{name}", "{address}", "{district}", "{is_open}")
```

Запрос добавляет все переданные значения в таблицу Кинотеатры.

7.6 Вставка зала

```
INSERT INTO rooms (cinema_id, number, capacity, category)
VALUES ("{str(cinema_id)}", "{str(number)}", "{str(capacity)}",
"{str(category)}")
```

Запрос добавляет все переданные значения в таблицу Залы.

7.7 Вставка фильма

```
INSERT INTO films (name, producer, operator, cost, country,
duration, picture)
VALUES ("{name}", "{prod}", "{operator}", "{str(cost)}",
"{country}", "{str(duration)}", "{pic}")
```

Запрос добавляет все переданные значения в таблицу Фильмы.

7.8 Вставка сеанса

```
INSERT INTO sessions (room_id, film_id, date_time, seats)
VALUES ("{str(room_id)}", "{str(film_id)}", "{date_time}",
"{str(seats)}")
```

Запрос добавляет все переданные значения в таблицу Сеансы.

7.9 Вставка жанра

```
INSERT INTO genres (film_id, genre)
VALUES ("{str(film_id)}", "{genre}")
```

Запрос добавляет все переданные значения в таблицу Жанры.

7.10 Вставка роли

```
INSERT INTO roles (name, surname, birth, film_id, role_name)
```

```
VALUES ("{name}", "{surname}", "{birth}", "{str(film_id)}",
"{role}")
```

Запрос добавляет все переданные значения в таблицу Роли.

7.11 Вставка приза

```
INSERT INTO prizes (name, year, nomination, film_id)
VALUES ("{name}", "{str(year)}", "{nomination}", "{str(film_id)}")
```

Запрос добавляет все переданные значения в таблицу Призы.

7.12 Удаление кинотеатра

Удаление кинотеатра должно повлечь за собой удаление залов этого кинотеатра и сеансов которые в этом зале проводятся или проводились. Поэтому используются три запроса.

```
DELETE FROM cinemas WHERE id="{str(cinema_id)}";
DELETE FROM rooms WHERE cinema_id="{str(cinema_id)}";
DELETE FROM sessions WHERE id IN ({",".join(map(str, sessions_id))});
```

7.13 Удаление фильма

Удаление фильма так же как и удаление кинотеатра должно повлекать за собой удаление ролей, жанров, и наград относящихся к данному фильму.

```
DELETE FROM films WHERE id="{str(film_id)}";
DELETE FROM roles WHERE film_id="{str(film_id)}";
DELETE FROM prizes WHERE film_id="{str(film_id)}";
DELETE FROM genres WHERE film_id="{str(film_id)}";
```

7.14 Удаление сеанса

```
DELETE FROM sessions WHERE id="{str(session_id)}";
```

Простое удаление записи из таблицы Сеансы.

7.15 Удаление роли

```
DELETE FROM roles WHERE id="{str(role_id)}";
```

Простое удаление записи из таблицы Роли.

7.16 Удаление актера

```
DELETE FROM roles WHERE name="{str(actor[0])}" AND
                           surname="{str(actor[1])}" AND
                           birth="{str(actor[2])}"
```

Удаление записей из таблицы Роли при заданном актере.

7.17 Удаление награды

```
DELETE FROM prizes WHERE id="{str(prize_id)}"
```

Удаление записей из таблицы Призы.

Заключение

В ходе выполнения данного индивидуального задания была реализована программная система для работников кинотеатров и справочной службы города.

Особенностью данной системы является четкое разделение пользователей по ролям, каждая из которых определяет доступный функционал.

Так, для работников кинотеатров доступно редактирования репертуара соответствующего кинотеатра и печать афиш на показываемые фильмы.

Для работников справочной службы предусмотрен более широкий функционал, а именно возможность добавления, изменения и удаления информации о кинотеатрах, актерах и фильмах в прокате, поиск сеансов по определенным фильтрам и печать отчета.

В работе использовался язык Python с библиотеками tkinter для графического пользовательского интерфейса и sqlite для работы с базами данных. С полным кодом работы можно ознакомиться в Github-репозитории по ссылке: https://github.com/snchz29/db_cinema.git

ПРИЛОЖЕНИЕ А

```
CREATE TABLE cinemas(  
    id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,  
    name TEXT(32) NOT NULL,  
    address TEXT(64) NOT NULL,  
    district TEXT(32) NOT NULL,  
    is_open INTEGER NOT NULL DEFAULT 1 CHECK ( is_open = 0 OR is_open = 1 )  
);
```

```
CREATE TABLE rooms(  
    id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,  
    cinema_id INTEGER NOT NULL,  
    number INTEGER NOT NULL CHECK ( number > 0 ),  
    capacity INTEGER NOT NULL CHECK ( capacity > 0 ),  
    category INTEGER NOT NULL CHECK ( category IN (1,2,3)),  
    FOREIGN KEY (cinema_id) REFERENCES cinemas(id)  
);
```

```
CREATE TABLE films(  
    id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,  
    name TEXT(32) NOT NULL,  
    producer TEXT(32) NOT NULL,  
    operator TEXT(32) NOT NULL,  
    cost INTEGER NOT NULL CHECK ( cost > 0 ),  
    country TEXT(32) NOT NULL,  
    duration INTEGER NOT NULL CHECK ( duration > 0 ),  
    picture TEXT(512) NOT NULL  
);
```

```
CREATE TABLE roles(  
    id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,  
    name TEXT(32) NOT NULL,  
    surname TEXT(32) NOT NULL,  
    birth TEXT(19) NOT NULL,  
    film_id INTEGER NOT NULL,  
    role_name TEXT(32) NOT NULL,  
    FOREIGN KEY (film_id) REFERENCES films(id)  
);
```

```
CREATE TABLE sessions(  
    id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,  
    room_id INTEGER NOT NULL,
```

```

        film_id INTEGER NOT NULL,
        date_time TEXT(19) NOT NULL,
        seats INTEGER NOT NULL check ( sessions.seats >= 0 ),
        FOREIGN KEY (room_id) REFERENCES rooms(id),
        FOREIGN KEY (film_id) REFERENCES films(id)
    );

CREATE TABLE prizes(
    id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
    name TEXT(32) NOT NULL,
    year INTEGER NOT NULL check ( year > 1900 AND year < 2050),
    nomination TEXT(32) NOT NULL,
    film_id INTEGER NOT NULL,
    FOREIGN KEY (film_id) REFERENCES films(id)
);

CREATE TABLE genres(
    id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
    film_id INTEGER NOT NULL,
    genre TEXT(16),
    FOREIGN KEY (film_id) REFERENCES films(id)
);

```

ПРИЛОЖЕНИЕ Б

```
INSERT INTO cinemas (name, address, district, is_open)
VALUES ("мир", "победы 1", "центральный", 1),
       ("победа", "пушкина 8", "приморский", 1),
       ("премьера", "ленина 88", "центральный", 1),
       ("эльдар", "ленина 256", "приморский", 1),
       ("искра", "пушкина 44", "центральный", 1);

INSERT INTO films (name, producer, operator, cost, country, duration, picture)
VALUES ("На игле", "Денни Бойл", "Брайан Тьюфани", 3000000, "Великобритания", 93,
       "https://avatars.mds.yandex.net/get-kinopoisk-image/1777765/f47dc447-7360-43a4-aa05-5ac408203f5e/300x450"),
       ("Страх и ненависть в Лас-Вегасе", "Терри Гиллиам", "Никола Пекорини",
       5000000, "США", 110,
       "https://b1.filmpro.ru/c/8866.jpg"),
       ("Джентельмены", "Гай Ричи", "Алан Стюарт", 2000000, "Великобритания", 113,
       "https://avatars.mds.yandex.net/get-kinopoisk-image/1599028/637271d5-61b4-4e46-ac83-6d07494c7645/600x900"),
       ("Карты, деньги, два ствола", "Гай Ричи", "Тим Морис-Джонс", 1000000,
       "Великобритания", 115,
       "https://b1.filmpro.ru/c/458171.jpg"),
       ("Криминальное чтиво", "Квентин Тарантино", "Анджей Секула", 2500000, "США",
       154,
       "https://upload.wikimedia.org/wikipedia/ru/9/93/Pulp_Fiction.jpg"),
       ("Интерстеллар", "Кристофер Нолан", "Хойте Ван Хойтема", 6000000, "США", 169,
       "https://avatars.mds.yandex.net/get-kinopoisk-image/1600647/430042eb-ee69-4818-aed0-a312400a26bf/300x450"),
       ("Золото", "Стивен Гейган", "Роберт Элсвит", 4000000, "США", 120,
       "https://avatars.mds.yandex.net/get-kinopoisk-image/1900788/0c30b502-4e7f-49ee-978c-383d05af3ff9/300x450");

INSERT INTO rooms (cinema_id, number, capacity, category)
VALUES (1, 1, 40, 1),
       (1, 2, 80, 2),
       (1, 3, 30, 1),
       (2, 1, 50, 3),
       (2, 2, 60, 3),
       (3, 1, 40, 1),
       (3, 2, 50, 3),
       (4, 1, 20, 2),
       (4, 2, 40, 1),
       (4, 3, 20, 2),
       (4, 4, 30, 2),
```

```
(5, 1, 40, 1),
(5, 2, 50, 1),
(5, 3, 60, 2),
(5, 4, 70, 3);
```

```
INSERT INTO sessions (room_id, film_id, date_time, seats)
```

```
VALUES (1, 1, "2021-01-10 12:00:00", 10),
(3, 2, "2021-01-10 12:00:00", 5),
(4, 2, "2021-01-10 13:00:00", 4),
(5, 3, "2021-01-10 13:00:00", 5),
(6, 3, "2021-01-11 13:00:00", 7),
(7, 4, "2021-01-11 14:00:00", 3),
(8, 5, "2021-01-11 14:00:00", 4),
(9, 1, "2021-01-11 14:00:00", 5),
(10, 1, "2021-01-11 15:00:00", 1),
(11, 3, "2021-01-10 15:00:00", 3),
(12, 5, "2021-01-10 15:00:00", 10),
(13, 2, "2021-01-10 16:00:00", 5),
(14, 4, "2021-01-10 16:00:00", 6),
(15, 2, "2021-01-10 16:00:00", 6),
(1, 5, "2021-01-11 17:00:00", 5),
(2, 5, "2021-01-11 17:00:00", 8),
(3, 5, "2021-01-11 17:00:00", 18),
(5, 5, "2021-01-11 18:00:00", 9),
(6, 5, "2021-01-10 18:00:00", 10),
(7, 5, "2021-01-10 19:00:00", 0),
(8, 1, "2021-01-10 19:00:00", 0),
(9, 3, "2021-01-10 19:00:00", 0),
(10, 2, "2021-01-10 20:00:00", 4),
(11, 2, "2021-01-10 20:00:00", 5),
(12, 2, "2021-01-10 20:00:00", 1),
(13, 2, "2021-01-10 21:00:00", 5),
(14, 4, "2021-01-10 21:00:00", 6),
(15, 4, "2021-01-10 21:00:00", 3);
```

```
INSERT INTO roles(name, surname, birth, film_id, role_name)
```

```
VALUES ("Юэн", "Макгрегор", "1989-10-01", 1, "Марк Рентон"),
("Роберт", "Карлайл", "1972-01-10", 1, "Френсис Бегби"),
("Джонни", "Депп", "1975-04-15", 2, "Рауль Дюк"),
("Бенисио", "дель Торо", "1969-11-05", 2, "Доктор Гонзо"),
("Чарли", "Ханнем", "1980-12-21", 3, "Реймонд Смит"),
("Мэттью", "Макконахи", "1970-14-05", 3, "Микки Пирсон"),
("Колин", "Фарелл", "1969-10-29", 3, "Тренер"),
("Джейсон", "Стейтем", "1962-10-07", 4, "Бейкон"),
```

```

("Джон", "Траволта", "1956-11-05", 5, "Винсент Вега"),
("Ума", "Турман", "1986-10-15", 5, "Мия Уоллес"),
("Сэмюэл", "Лерой Джексон", "1946-12-27", 5, "Джулс Уиннфилд"),
("Брюс", "Уиллис", "1966-01-02", 5, "Бутч Куллидж"),
("Тим", "Рот", "1980-01-20", 5, "Тыковка"),
("Мэттью", "Макконахи", "1970-14-05", 6, "Купер"),
("Энн", "Хэтүэй", "1985-10-24", 6, "Амелия Бренд"),
("Мэттью", "Макконахи", "1970-14-05", 7, "Кенни Уэллс");

INSERT INTO prizes(name, year, nomination, film_id)
VALUES ("Премия БАФТА", 1996, "Лучший адаптированный сценарий", 1),
("Золотая пальмовая ветвь", 1998, "", 2),
("Премия БАФТА", 1999, "Выдающийся британский фильм года", 4),
("Премия британского независимого кино", 1998, "Лучший британский независимый фильм", 4),
("Золотая пальмовая ветвь", 1994, "", 5),
("Оскар", 1995, "лучший оригинальный сценарий", 5),
("Оскар", 2015, "лучшие визуальные эффекты", 6),
("Империя", 2015, "лучший фильм", 6);

INSERT INTO genres(genre, film_id)
VALUES ("Комедия", 1),
("Драма", 1),
("Криминал", 1),
("Комедия", 2),
("Сатира", 2),
("Приключения", 2),
("Драма", 2),
("Боевик", 3),
("Комедия", 3),
("Криминал", 3),
("Комедия", 4),
("Триллер", 4),
("Криминал", 4),
("Комедия", 5),
("Криминал", 5),
("Триллер", 5),
("Драма", 5),
("Фантастика", 6),
("Приключения", 6),
("Драма", 6),
("Детектив", 6),
("Приключения", 7),
("Триллер", 7),

```

```
("Драма", 7),  
("Криминал", 7);
```

ПРИЛОЖЕНИЕ В

```
import sqlite3
import functools

DATABASE_PATH = 'new_data.sqlite'

def find_best_id(ids):
    id_ = max(ids) + 1
    for i in range(1, max(ids) + 2):
        if i not in ids:
            id_ = i
            break
    return id_

def getter(decorated):
    @functools.wraps(decorated)
    def wrapper(*args, **kwargs):
        conn = sqlite3.connect(DATABASE_PATH)
        cursor = conn.cursor()
        cursor.execute(decorated(*args, **kwargs))
        res = cursor.fetchall()
        conn.close()
        return res
    return wrapper

def worker(decorated):
    @functools.wraps(decorated)
    def wrapper(*args, **kwargs):
        conn = sqlite3.connect(DATABASE_PATH)
        cursor = conn.cursor()
        cursor.execute(decorated(*args, **kwargs))
        conn.commit()
        conn.close()
    return wrapper

class DbHolder:
    @staticmethod
    @getter
    def get(table: str):
        return f'SELECT * FROM {table}'

    @staticmethod
    @getter
    def get_cinema_by_id(cinema_id):
        return f'SELECT * FROM cinemas WHERE id="{cinema_id}"'

    @staticmethod
    @getter
    def get_sessions_by_cinema(cinema_id):
```

```

        return f'''SELECT sessions.id, rooms.number, name, date_time, films.cost / 10000 * (4-
rooms.category) AS cost, seats
        FROM sessions INNER JOIN films ON film_id = films.id
        INNER JOIN rooms ON room_id = rooms.id
        WHERE room_id IN (SELECT id
                        FROM rooms
                        WHERE cinema_id="{cinema_id}" OR cinema_id={cinema_id});'''

    @staticmethod
    @getter
    def get_sessions_by_film(film_id):
        return f'''SELECT sessions.id, cinemas.name, rooms.number, date_time, seats,
                        films.cost / 10000 * (4-rooms.category) AS cost
        FROM sessions INNER JOIN films ON film_id = films.id
        INNER JOIN rooms ON room_id = rooms.id
        INNER JOIN cinemas ON rooms.cinema_id = cinemas.id
        WHERE film_id = "{film_id}" OR film_id = {film_id};'''

    @staticmethod
    @getter
    def get_sessions_by_film_and_cinema(film_id, cinema_id):
        return f'''SELECT sessions.id, cinemas.name, rooms.number, date_time, seats,
                        films.cost / 10000 * (4-rooms.category) AS cost
        FROM sessions INNER JOIN films ON film_id = films.id
        INNER JOIN rooms ON room_id = rooms.id
        INNER JOIN cinemas ON rooms.cinema_id = cinemas.id
        WHERE (film_id = "{film_id}" OR film_id = {film_id}) AND
                (cinemas.id = "{cinema_id}" OR cinemas.id = {cinema_id});'''

    @staticmethod
    @getter
    def get_sessions_by_actor(actor):
        return f'''SELECT sessions.id, films.name, cinemas.name, rooms.number, date_time, seats,
                        films.cost / 10000 * (4-rooms.category) AS cost
        FROM sessions INNER JOIN films ON sessions.film_id = films.id
        INNER JOIN roles ON roles.film_id = films.id
        INNER JOIN rooms ON room_id = rooms.id
        INNER JOIN cinemas ON rooms.cinema_id = cinemas.id
        WHERE roles.name = "{actor[0]}" AND roles.surname = "{actor[1]}" AND roles.birth =
"{actor[2]}";'''

    @staticmethod
    @getter
    def get_sessions_by_date(date, time_b, time_e, genre):
        return f'''SELECT sessions.id, f.name, c.name, r.number, date_time, seats,
                        f.cost / 10000 * (4-r.category) AS cost, GROUP_CONCAT(g.genre)
        FROM sessions INNER JOIN rooms r ON r.id = sessions.room_id
        INNER JOIN cinemas c ON c.id = r.cinema_id
        INNER JOIN films f ON f.id = sessions.film_id AND "{genre}" IN (SELECT genre
                                                                    FROM genres
                                                                    WHERE f.id =
genres.film_id)
        INNER JOIN genres g ON f.id = g.film_id
        GROUP BY sessions.id
        HAVING date_time > "{date} {time_b}" AND date_time < "{date} {time_e}"""'''

```



```

    @staticmethod
    @getter
    def get_room_by_id(room_ids):
        return f'''SELECT * FROM rooms WHERE id IN ({",".join("'" + str(i) + "'" for i in
room_ids}})'''

    @staticmethod
    @getter
    def get_rooms_by_cinema(cinema_id):
        return f'''SELECT id, number, capacity FROM rooms WHERE cinema_id="{str(cinema_id)}"'''

    @staticmethod
    @getter
    def get_films_by_id(film_id):
        return f'''SELECT * FROM films WHERE id="{film_id}" OR id={film_id}'''

    @staticmethod
    @getter
    def get_film_id_by_name(name, prod):
        return f'''SELECT id FROM films WHERE name="{name}" and producer="{prod}"'''

    @staticmethod
    @getter
    def get_films_by_cinema(cinema_id):
        return f'''SELECT films.id, films.name, films.producer, films.operator, films.country,
films.duration,
                GROUP_CONCAT(DISTINCT genres.genre), films.picture
        FROM films INNER JOIN sessions ON sessions.film_id = films.id
                INNER JOIN rooms ON rooms.id = sessions.room_id AND
rooms.cinema_id="{cinema_id}"
                INNER JOIN genres ON films.id = genres.film_id
        GROUP BY films.id;'''

    @staticmethod
    @getter
    def get_genres_by_film_id(film_id):
        return f'''SELECT genre FROM genres WHERE film_id="{str(film_id)}"'''

    @staticmethod
    @getter
    def get_roles_by_film_id(film_id):
        return f'''SELECT * FROM roles WHERE film_id="{str(film_id)}"'''

    @staticmethod
    @getter
    def get_actors():
        return '''SELECT DISTINCT name, surname, birth FROM roles;'''

    @staticmethod
    @getter
    def get_actors_by_film_id(film_id):
        return f'''SELECT name, surname, role_name
        FROM roles
        WHERE film_id = "{str(film_id)}";'''

```

```

@staticmethod
@getter
def get_prizes_by_film_id(film_id):
    return f'''SELECT * FROM prizes WHERE prizes.film_id="{str(film_id)}"'''

@staticmethod
@getter
def get_report_by_cinema_films():
    return '''SELECT c.district, c.name, f.name, avg((4 - r.category) * f.cost/10000) AS avg_cost,
count(s.id)
FROM films INNER JOIN sessions s ON films.id = s.film_id
INNER JOIN rooms r ON s.room_id = r.id
INNER JOIN cinemas c ON c.id = r.cinema_id
INNER JOIN films f ON f.id = s.film_id
GROUP BY cinema_id, film_id
ORDER BY district, c.name, f.name;'''

@staticmethod
@getter
def get_report_cinemas_income():
    return '''SELECT sum(income), cinema, district
FROM (SELECT c.name as cinema, c.district as district,
sum(r.capacity - sessions.seats) * avg((4 - r.category)*f.cost/10000) -
f.cost as income
FROM sessions INNER JOIN rooms r ON r.id = sessions.room_id
INNER JOIN films f ON f.id = sessions.film_id
INNER JOIN cinemas c ON c.id = r.cinema_id
GROUP BY film_id, r.cinema_id
ORDER BY r.cinema_id)
GROUP BY cinema
ORDER BY district, cinema;'''

@staticmethod
@getter
def get_report_district_income():
    return '''SELECT district, sum(income)
FROM (SELECT sum(income) AS income, district
FROM (SELECT c.name AS cinema, c.district AS district,
sum(r.capacity-sessions.seats) * avg(f.cost/10000 * (4 - r.category)) -
f.cost AS income
FROM sessions INNER JOIN rooms r ON r.id = sessions.room_id
INNER JOIN films f ON f.id = sessions.film_id
INNER JOIN cinemas c ON c.id = r.cinema_id
GROUP BY film_id, r.cinema_id
ORDER BY r.cinema_id)
GROUP BY cinema)
GROUP BY district;'''

@staticmethod
@getter
def get_report_full_income():
    return '''SELECT SUM(income)
FROM (SELECT district, sum(income) AS income
FROM (SELECT sum(income) AS income, district

```

```

        FROM (SELECT c.name AS cinema, c.district AS district,
                    sum(r.capacity-sessions.seats) * avg(f.cost/10000 * (4 - r.category)) -
f.cost AS income
                FROM sessions INNER JOIN rooms r ON r.id = sessions.room_id
                INNER JOIN films f ON f.id = sessions.film_id
                INNER JOIN cinemas c ON c.id = r.cinema_id
                GROUP BY film_id, r.cinema_id
                ORDER BY r.cinema_id)
        GROUP BY cinema)
    GROUP BY district);'''

@staticmethod
@getter
def get_report_genres():
    return '''SELECT genre, count(film_id) FROM genres GROUP BY genre;'''

@staticmethod
@worker
def update_cinemas(cinema_id, name, address, district, is_open):
    return f'''UPDATE cinemas
                SET name="{name}", address="{address}", district="{district}", is_open="{is_open}"
                WHERE id="{cinema_id}'''

@staticmethod
@worker
def update_session(session_id, room_id, date_time, seats):
    return f'''UPDATE sessions
                SET room_id="{room_id}", date_time="{date_time}", seats="{seats}"
                WHERE id="{session_id}'''

@staticmethod
@worker
def update_film(film_id, name, prod=None, operator=None, cost=None, country=None, duration=None,
pic=None):
    if prod is None:
        return f'''UPDATE films SET name="{name}" WHERE id="{str(film_id)}'''
    else:
        return f'''UPDATE films
                    SET name="{name}", producer="{prod}", operator="{operator}", cost="{str(cost)}",
                    country="{country}", duration="{str(duration)}", picture="{pic}"
                    WHERE id="{str(film_id)}'''

@staticmethod
@worker
def update_actor(old, name, surname, birth):
    return f'''UPDATE roles
                SET name="{name}", surname="{surname}", birth="{birth}"
                WHERE name="{old[0]}" AND surname="{old[1]}" AND birth="{old[2]}'''

@staticmethod
@worker
def insert_cinema(name, address, district, is_open):
    return f'''INSERT INTO cinemas (name, address, district, is_open)
                VALUES ("{name}", "{address}", "{district}", "{is_open}")'''

```

```

@staticmethod
@worker
def insert_room(cinema_id, number, capacity, category):

    return f'''INSERT INTO rooms (cinema_id, number, capacity, category)
            VALUES ("{str(cinema_id)}", "{str(number)}", "{str(capacity)}", "{str(category)}")'''

@staticmethod
@worker
def insert_film(name, prod, operator, cost, country, duration, pic):
    return f'''INSERT INTO films (name, producer, operator, cost, country, duration, picture)
            VALUES ("{name}", "{prod}", "{operator}", "{str(cost)}", "{country}",
                    "{str(duration)}", "{pic}")'''

@staticmethod
@worker
def insert_session(room_id, film_id, date_time, seats):
    return f'''INSERT INTO sessions (room_id, film_id, date_time, seats)
            VALUES ("{str(room_id)}", "{str(film_id)}", "{date_time}", "{str(seats)}")'''

@staticmethod
@worker
def insert_genre(film_id, genre):
    return f'''INSERT INTO genres (film_id, genre)
            VALUES ("{str(film_id)}", "{genre}")'''

@staticmethod
@worker
def insert_role(film_id, name, surname, birth, role):
    return f'''INSERT INTO roles (name, surname, birth, film_id, role_name)
            VALUES ("{name}", "{surname}", "{birth}", "{str(film_id)}", "{role}")'''

@staticmethod
@worker
def insert_prize(film_id, name, year, nomination):
    return f'''INSERT INTO prizes (name, year, nomination, film_id)
            VALUES ("{name}", "{str(year)}", "{nomination}", "{str(film_id)}")'''

def delete_cinema(self, cinema_id):
    sessions_id = [s[0] for s in self.get_sessions_by_cinema(cinema_id)]
    self._delete_cinema_from_cinemas(cinema_id)
    self._delete_cinema_from_rooms(cinema_id)
    self._delete_cinema_from_sessions(sessions_id)

@staticmethod
@worker
def _delete_cinema_from_cinemas(cinema_id):
    return f'''DELETE FROM cinemas WHERE id="{str(cinema_id)}"'''

@staticmethod
@worker
def _delete_cinema_from_rooms(cinema_id):
    return f'''DELETE FROM rooms WHERE cinema_id="{str(cinema_id)}"'''

@staticmethod

```

```

@worker
def _delete_cinema_from_sessions(sessions_id):
    return f'''DELETE FROM sessions WHERE id IN ({",".join(map(str, sessions_id))})'''

def delete_film(self, film_id):
    self._delete_film_from_films(film_id)
    self._delete_film_from_roles(film_id)
    self.delete_genre(film_id)
    self._delete_film_from_prizes(film_id)

@staticmethod
@worker
def _delete_film_from_films(film_id):
    return f'''DELETE FROM films WHERE id="{str(film_id)}";'''

@staticmethod
@worker
def _delete_film_from_roles(film_id):
    return f'''DELETE FROM roles WHERE film_id="{str(film_id)}";'''

@staticmethod
@worker
def _delete_film_from_prizes(film_id):
    return f'''DELETE FROM prizes WHERE film_id="{str(film_id)}"'''

@staticmethod
@worker
def delete_session(session_id):
    return f'''DELETE FROM sessions WHERE id="{str(session_id)}"'''

@staticmethod
@worker
def delete_genre(film_id):
    return f'''DELETE FROM genres WHERE film_id="{str(film_id)}"'''

@staticmethod
@worker
def delete_role(role_id):
    return f'''DELETE FROM roles WHERE id="{str(role_id)}"'''

@staticmethod
@worker
def delete_actor(actor):
    return f'''DELETE FROM roles WHERE name="{str(actor[0])}" AND
        surname="{str(actor[1])}" AND
        birth="{str(actor[2])}"'''

@staticmethod
@worker
def delete_prize(prize_id):
    return f'''DELETE FROM prizes WHERE id="{str(prize_id)}"'''

```