

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ  
УНИВЕРСИТЕТ»**

**ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ**

**ОТЧЕТ**

**по реализации приложения, взаимодействующего с базами данных средствами  
языков SQL, PL/SQL по курсу “Базы данных” ФИТ НГУ**

**Обучающегося:      Кубышевой Татьяны Алексеевны группы № 19212 курса 3**

**Тема задания:      Схема базы данных театра**

Новосибирск 2022

<b>Содержание</b>	
<b>Введение</b>	<b>3</b>
<b>Глава 1. Анализ проектного задания</b>	<b>5</b>
<b>Основные сущности и отношения</b>	<b>5</b>
Работник театра	5
Спектакль	5
Репертуар	5
Билет	5
Абонемент	5
Касса	5
<b>Ограничения целостности</b>	<b>6</b>
Ограничения целостности на работников	6
Ограничения целостности на спектакли, репертуар, авторов, возрастную категорию, жанры, билеты, подписку по жанрам и по авторам	6
Ограничения целостности на спектакль	6
<b>Роли пользователей приложения</b>	<b>6</b>
Администратор базы данных	7
Администратор театра	7
Директор	7
Посетитель	7
<b>Глава 2. Проектирование системы</b>	<b>8</b>
<b>Архитектура</b>	<b>8</b>
<b>Обеспечение ограничений целостности</b>	<b>8</b>
<b>Авторизация</b>	<b>8</b>
<b>Построение интерфейса в соответствии с ролями пользователей</b>	<b>8</b>
<b>Глава 3. Реализации системы</b>	<b>9</b>
<b>Клиентская программа</b>	<b>9</b>
<b>Подключение к базе данных</b>	<b>9</b>
<b>Несколько примеров форм приложения</b>	<b>10</b>
Начало работы	10
Основное меню	12
Работа с таблицами	13

Работа с отчетами	13
Параметры отчетов	14
<b>Сервер СУБД</b>	<b>14</b>
<b>Глава 4. Тестирование и эксплуатация</b>	<b>23</b>
<b>Заключение</b>	<b>23</b>

## Введение

Работников театра можно подразделить на актеров, музыкантов, постановщиков и служащих. Каждая из перечисленных категорий имеет уникальные атрибуты-характеристики и может подразделяться (например, постановщики) на более мелкие категории. Театр возглавляет директор, в функции которого входят контроль за постановками спектаклей, утверждение репертуара, принятие на работу новых служащих, приглашение актеров и постановщиков. Актеры, музыканты и постановщики, работающие в театре, могут уезжать на гастроли. Актеры театра могут иметь звания заслуженных и народных артистов, могут быть лауреатами конкурсов. Также актерами театра могут быть и студенты театральных училищ. Каждый актер имеет свои вокальные и внешние данные (пол, возраст, голос, рост и т.п.), которые могут подходить для каких-то ролей, а для каких-то нет (не всегда женщина может сыграть мужчину и наоборот).

Для постановки любого спектакля необходимо подобрать актеров на роли и дублеров на каждую главную роль. Естественно, что один и тот же актер не может играть более одной роли в спектакле, но может играть несколько ролей в различных спектаклях. У спектакля также имеется режиссер-постановщик, художник-постановщик, дирижер-постановщик, авто.

Спектакли можно подразделить по жанрам: музыкальная комедия, трагедия, оперетта и пр. С другой стороны, спектакли можно подразделить на детские, молодежные и пр. В репертуаре театра указывается какие спектакли, в какие дни и в какое время будут проходить, а также даты премьер. В кассах театра можно заранее приобрести билеты или абонемент на любые спектакли. Абонемент обычно включает в себя билеты на спектакли либо конкретного автора, либо конкретного жанра. Цена билетов зависит от места, и спектакля. На премьеры билеты дороже. Администрацией театра фиксируется количество проданных билетов на каждый спектакль.

Необходимо:

- Разработать СУБД
- Разработать интерфейс пользователя и сервера БД
- Произвести тестирование по эксплуатации приложения

## Глава 1. Анализ проектного задания

### Основные сущности и отношения

#### Работник театра

Данная сущность описывает работников театра, которые подразделяются на актеров, музыкантов, постановщиков и прочих. Для каждой роли заведена отдельная сущность, так как у каждой из них имеются индивидуальные характеристики. Работники участвуют в создании спектаклей.

#### Спектакль

Данная сущность описывает все существующие на данный момент спектакли, которые показывались раньше, и планируют показываться в будущем. Спектакли ставятся на произведения авторов, организуют их постановщики, для каждого спектакля определяется жанр и возрастная категория. Спектакли далее заносятся в репертуар администратором.

#### Репертуар

Данная сущность описывает актуальное расписание спектаклей. Просмотр этой сущности доступен всем типам пользователей. Редактируется администратором театра.

#### Билет

Данная сущность описывает приобретенные билеты, на них указаны место, ряд, цена билета. При покупке билета добавляется новая строка.

#### Аbonемент

Данная сущность описывает приобретенные абонементы. Они бывают двух типов: на посещение спектаклей одного жанра или одного автора. При покупке в таблицу добавляется новая строка.

#### Касса

Данная сущность описывает совершенную покупку билета или абонемента, а также дату и время покупки.

Ниже представлена ER-диаграмма основных сущностей.

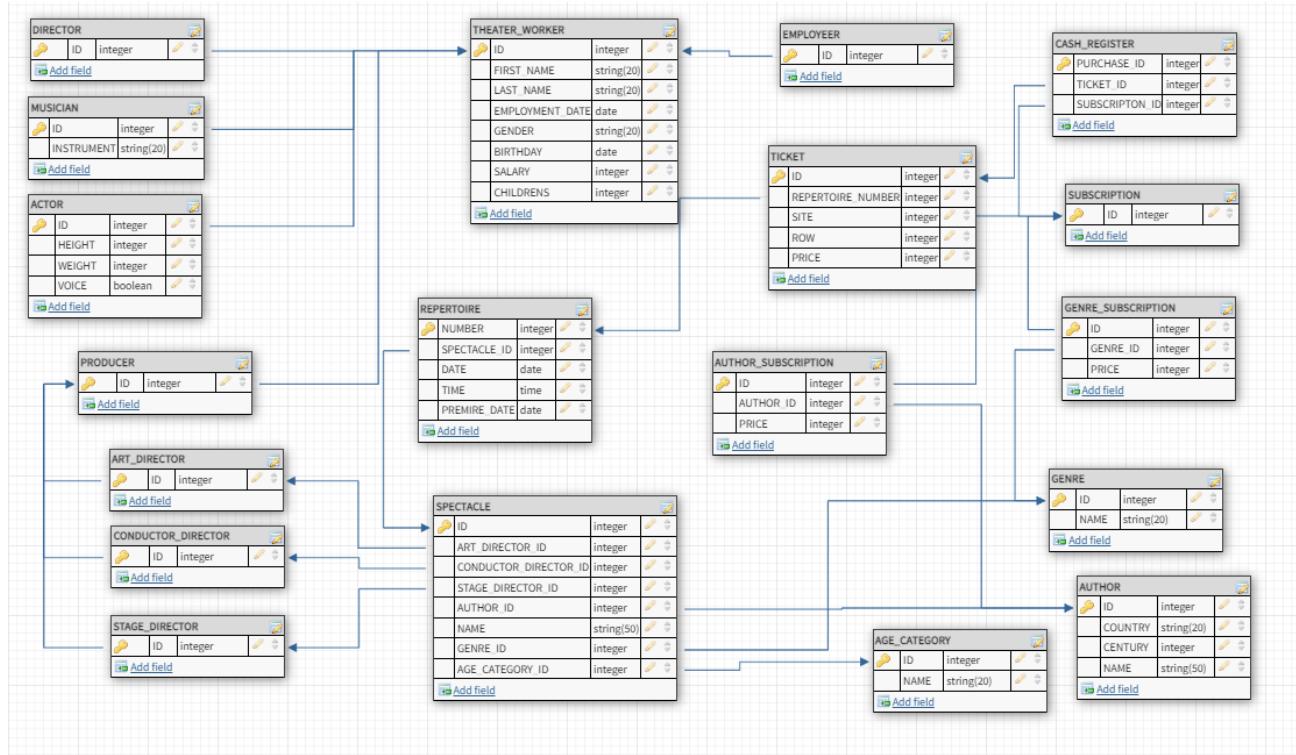


Рисунок 1 Диаграмма сущностей

## Ограничения целостности

### Ограничения целостности на работников

Каждый работник должен иметь уникальный идентификатор (он заполняется автоматически с помощью триггера), так как он используется при распределении по должностям в дочерние таблицы, и по нему определяются поля работников в таблице спектаклей.

### Ограничения целостности на спектакли, репертуар, авторов, возрастную категорию, жанры, билеты, подписку по жанрам и по авторам

Каждая из этих сущностей имеет индивидуальный идентификатор, определяемый автоматически.

### Ограничения целостности на спектакль

Каждый спектакль имеет трех разных постановщиков, автора, жанр и категорию.

## Роли пользователей приложения

Всего выделено три категории:

1. Администратор БД
2. Администратор театра
3. Директор
4. Посетитель

### Администратор базы данных

Администратор базы данных имеет доступ ко всей базе данных, он может добавлять новые жанры, новых авторов, редактировать все существующие таблицы.

### Администратор театра

Администратор театра управляет репертуаром и имеет сведения о проданных билетах.

### Директор

Директор может принимать на работу новых работников: актеров, музыкантов и тд., увольнять сотрудников, составлять новые спектакли и назначать тех, кто над ними будет работать, убирать спектакли из списка доступных. Имеет статистику по работникам и спектаклям.

### Посетитель

Посетитель может только просматривать репертуар, покупать билеты и абонементы.

Диаграмма вариантов представлена на рисунке ниже.

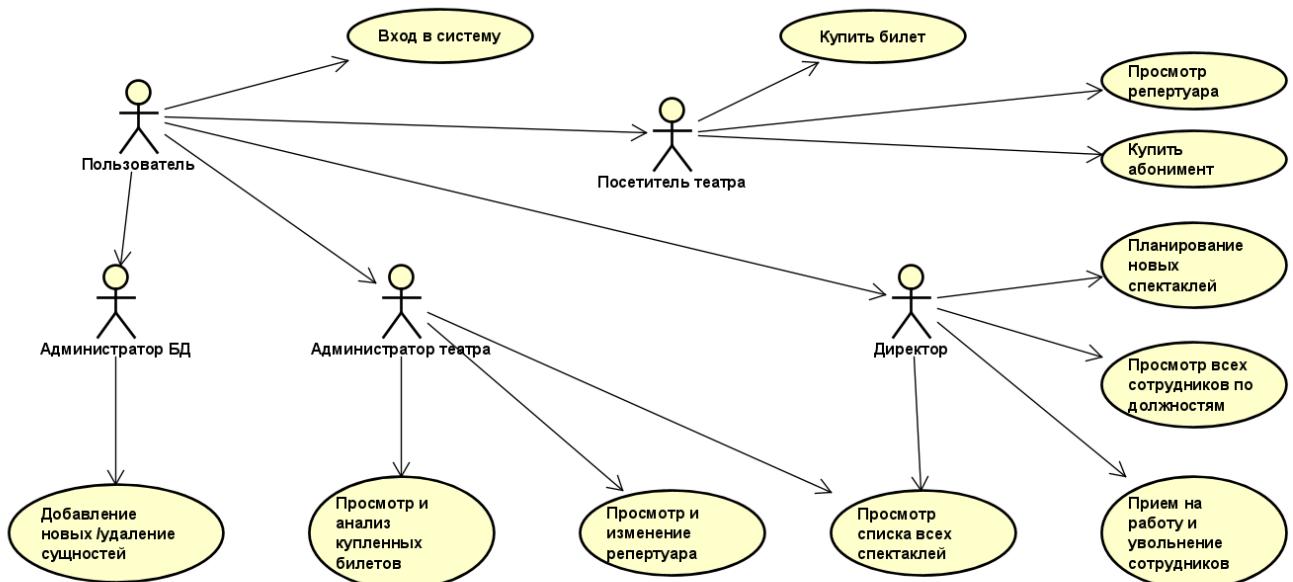


Рисунок 2 Диаграмма вариантов использования

## Глава 2. Проектирование системы

### Архитектура

Чтобы разделить роли, было принято решение менять интерфейс приложения в зависимости от роли пользователя. В зависимости от этой роли пользователю предоставляется доступ к окну с возможными ему функциональностями.

Приложение имеет клиент-серверную архитектуру: программа пользователя получает данные с удаленного сервера с СУБД, а также позволяет пользователю эти данные редактировать.

### Обеспечение ограничений целостности

Обеспечение ограничений целостности производится на нескольких уровнях.

1. Каждая таблица имеет первичный ключ
2. Связь между таблицами обеспечивается с помощью внешних ключей, также соблюдается каскадное обновление и удаление.
3. Oracle не имеет логических типов, поэтому было решено использовать строки ‘yes’/‘no’.

Обеспечение целостности с помощью **триггеров** используется для автоматической инкрементации при создании новых записей в таблицах.

### Авторизация

При входе в приложение пользователь выбирает под какой ролью он хочет войти в приложение, затем он вводит свой логин и пароль, после чего система проверяет наличие такого пользователя в базе и либо открывает ему вход в приложение с теми правами, которые у него есть, либо выводит сообщение об ошибке.

Имеется 4 уровня доступа:

Уровень администратора БД

Уровень администратора театра

Уровень директора

Уровень посетителя

В зависимости от роли открывается соответствующее меню.

### Построение интерфейса в соответствии солями пользователей

Ниже будет рассмотрено распределение интерфейсных окон по солям пользователей.

Интерфейс	Посетитель	Админ театра	Директор	Админ БД
Окно входа				
Форма “Работник театра” и дочерние				
Форма “Репертуар”				
Форма “Спектакль”				
Форма “Билет”				
Форма “Подписка”				
Форма “Автор”				
Форма “Жанр”				
Форма “Касса”				



## Глава 3. Реализации системы

### Клиентская программа

При реализации системы было построено приложение на языке Java с помощью графической библиотеки javafx. Интерфейс приложения меняется в зависимости от прав доступа, которые присваиваются когда пользователь выбирает под каким профилем ему зайти. На каждой форме есть кнопки, открывающие другие интерфейсные окна.

### Подключение к базе данных

Для подключения к базе данных был создан класс ConnectionDriver, который выполняет подключение к удаленному серверу по заданному адресу, с указанными логином и паролем:

```

public ConnectionDriver() throws ClassNotFoundException {
    Class.forName("oracle.jdbc.driver.OracleDriver");
}

public boolean connect(String ip, String login, String password) throws SQLException {
    Locale.setDefault(Locale.ENGLISH);
    String url = "jdbc:oracle:thin:@"+ip+":1521:XE";
    connection = DriverManager.getConnection(url, login, password);
    return connection != null;
}

```

Для работы с добавлением и редактированием полей таблиц созданы классы TheaterWorkerMenu, SpectacleMenu и тд.

Для работы с запросами SQL используются методы executeQuery и execute. Методы, которыми пользовались для получения/изменения информации

```

public ResultSet getTheaterWorker() throws SQLException {
    statement = connection.createStatement();
    return statement.executeQuery(sql: """
        |SELECT * FROM THEATER_WORKER""");
}

```

```

public void updateTheaterWorker(String firstName, String lastName, int salary, String birthdate)
public void deleteTheaterWorker(int id) throws SQLException {
public void commitTheaterWorker(String firstName, String lastName, int salary, String birthdate)

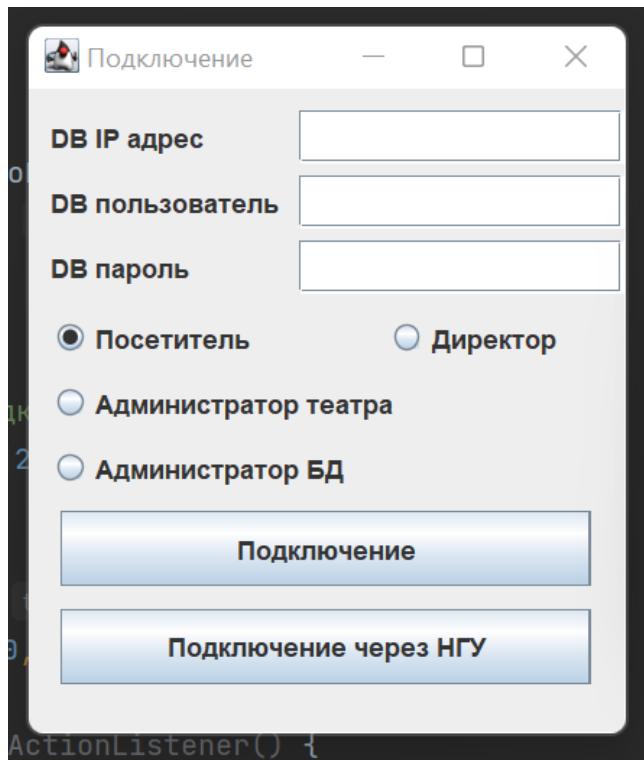
```

Для всех форм данные функции одинаковые.

## Несколько примеров форм приложения

### Начало работы

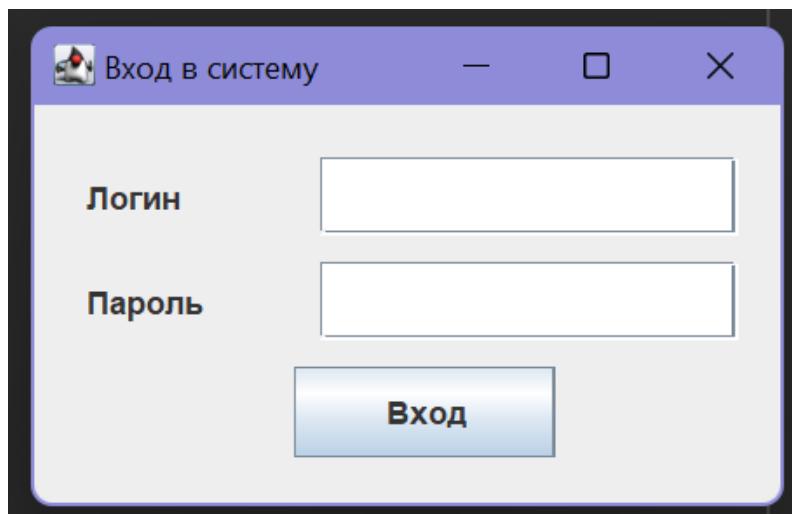
При запуске приложения открывается следующее окно:



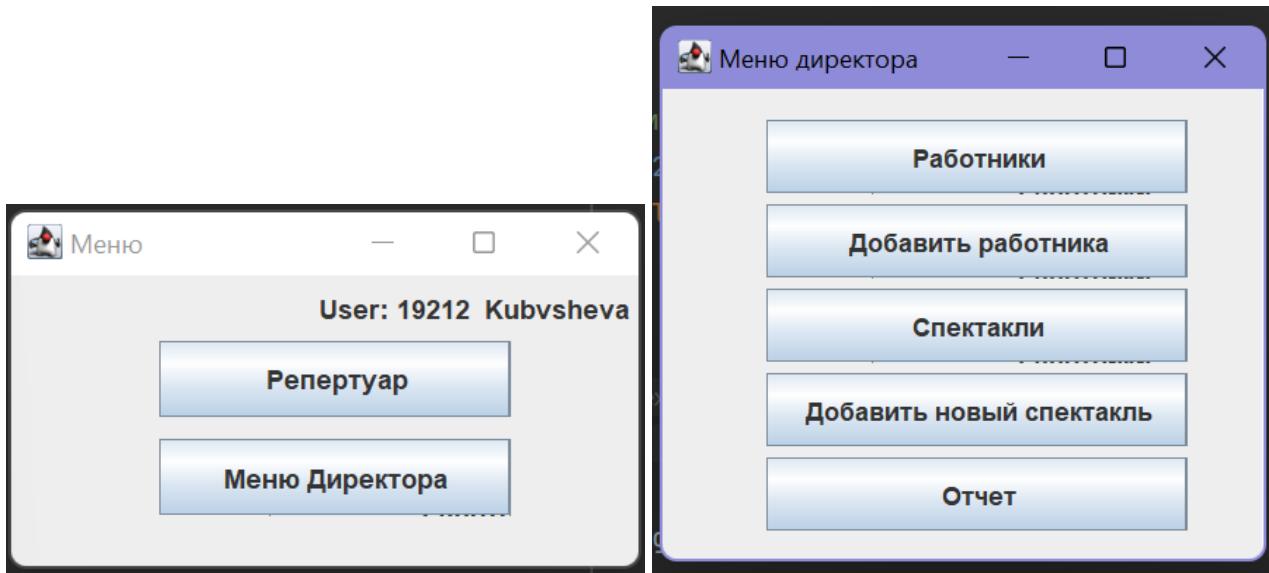
Здесь доступно несколько способов подключиться к базе данных:

1. Сервер курса НГУ под аккаунтом исполнителя задания (Вход по нажатию кнопки «Подключение через НГУ»)
2. Вход с пользовательскими настройками, с предварительным вводом параметров: IP адреса, логина и пароля.

В качестве логина вводится идентификатор пользователя и пароль. Все пароли хранятся в property файле на стороне сервера. Как только пользователь заполнит все поля и нажмет “Вход”, то выполняется проверка корректности, и после успешной проверки приложение открывает основное меню с возможностью перехода на другие формы.



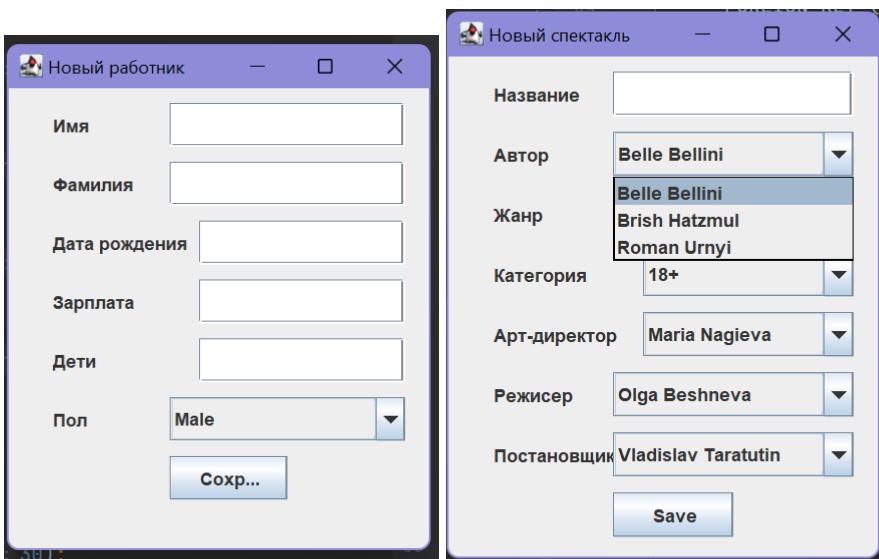
## Основное меню



В данном окне представлено две кнопки: кнопка, открывающая окно с репертуаром, общая для всех пользователей, а также индивидуальная кнопка для каждой роли. В индивидуальном меню расположены основные функции данной роли и кнопка “Отчет”, ведущая к реализации требований задания.

## Работа с таблицами

Фамилия	Имя	Дата трудоустройства	Пол	Дата рождения	Зарплата	Дети
Anna	Lomova	2022-05-14 14:54...	Female	2000-10-10 00:00...	30000	0
Sam	Lover	2022-05-14 15:00...	Male	1999-10-10 00:00...	12000	1
Bogdanov	Stanislav	2007-05-01 00:00...	Male	1985-09-07 00:00...	30000	0
Pechkin	Petr	2006-04-10 00:00...	Male	1961-11-04 00:00...	51000	3
Shubina	Anastasia	2011-01-20 00:00...	Female	1995-04-11 00:00...	41000	1
Ostapchenko	Ivan	2013-03-03 00:00...	Male	1986-12-01 00:00...	32000	1
Bogdanova	Sofia	2007-05-01 00:00...	Female	1985-09-07 00:00...	30000	0
Nagieva	Maria	2018-03-23 00:00...	Female	1999-11-03 00:00...	39000	0
Taratutin	Vladislav	2020-02-08 00:00...	Male	1988-06-07 00:00...	45000	1
Yatmanov	Oleg	2010-04-12 00:00...	Male	1995-04-10 00:00...	52000	0
Beshneva	Olga	1985-06-24 00:00...	Female	1965-10-11 00:00...	40000	1
Krasnoverova	Alisa	2009-04-23 00:00...	Female	1978-09-07 00:00...	55000	2
Bessonov	Pavel	2012-04-22 00:00...	Male	1983-03-03 00:00...	47000	1
Bessonova	Polina	2015-08-24 00:00...	Female	1985-12-05 00:00...	50000	1

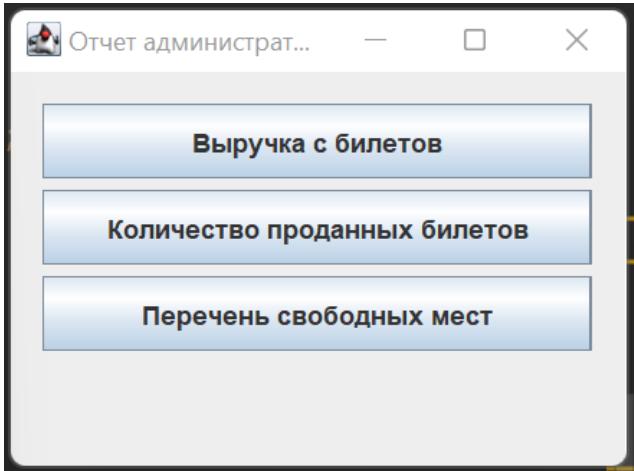


На большей части экрана размещены строки таблицы, которые можно просматривать. Снизу расположена панель с кнопками. С помощью кнопок можно манипулировать данными в таблице.

Добавление данных в таблицу предельно упрощена, большую часть полей нужно выбирать из уже предложенных.

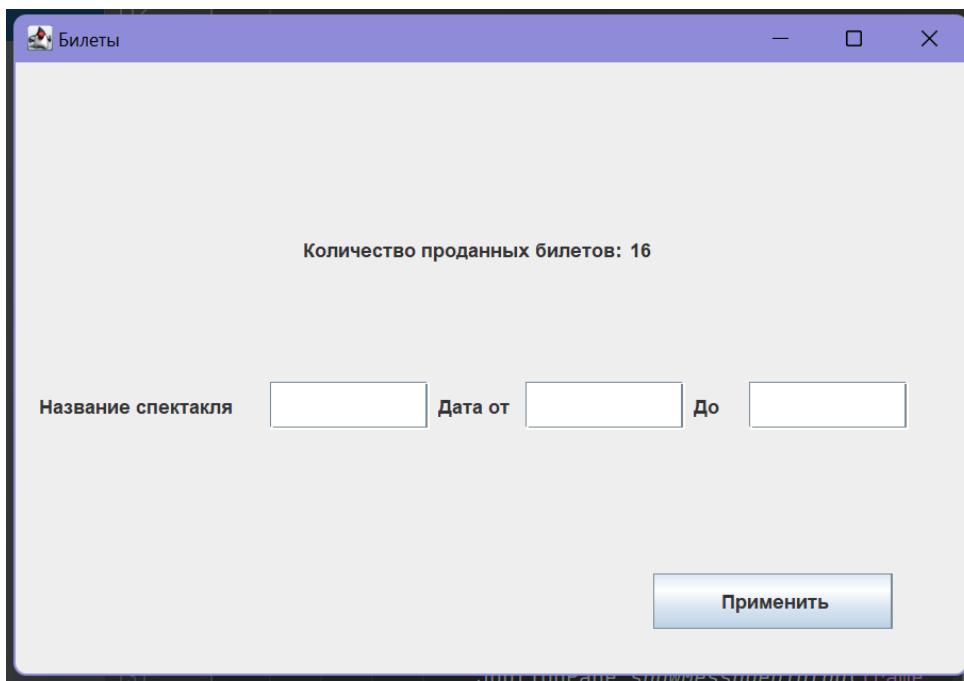
## Работа с отчетами

Отчеты открываются по нажатию кнопки «Отчет» в индивидуальном меню. После нажатия откроется диалоговое окно с несколькими отчетами, в зависимости от роли пользователя. Пример представлен ниже:



## Параметры отчетов

Большинство запросов системы параметрические, поэтому для них предусмотрены формы, где можно задать параметры поиска.



## Сервер СУБД

Приведем несколько примеров скриптов, которые используются при создании самой схемы базы данных:

```

1.
2. CREATE TABLE "THEATER_WORKER"
3.  (   "ID" NUMBER,
4.      "FIRST_NAME" VARCHAR2(20) NOT NULL ENABLE,
5.      "LAST_NAME" VARCHAR2(20) NOT NULL ENABLE,
6.      "EMPLOYMENT_DAY" DATE NOT NULL ENABLE,
7.      "GENDER" VARCHAR2(20),
8.      "BIRTHDAY" DATE,
9.      "RANK" VARCHAR2(20),
10.     "SALARY" NUMBER NOT NULL ENABLE,
11.     "CHILDRENS" NUMBER,
12.     PRIMARY KEY ("ID") ENABLE
13. );
14.
15.CREATE TABLE "PRODUCER"
16.  ( "ID" NUMBER,
17.      PRIMARY KEY ("ID") ENABLE,
18.      FOREIGN KEY ("ID")
19.        REFERENCES "THEATER_WORKER" ("ID") ENABLE
20. );
21.
22.CREATE TABLE "MUSICIAN"
23.  ( "ID" NUMBER,
24.      "INSTRUMENT" VARCHAR2(20),
25.      PRIMARY KEY ("ID") ENABLE,
26.      FOREIGN KEY ("ID")
27.        REFERENCES "THEATER_WORKER" ("ID") ENABLE

```

```
28.    );
29.
30.CREATE TABLE "ACTOR"
31. (    "ID" NUMBER,
32.     "HEIGHT" NUMBER,
33.     "WEIGHT" NUMBER,
34.     "VOICE" VARCHAR2(3),
35.         PRIMARY KEY ("ID") ENABLE,
36.         FOREIGN KEY ("ID")
37.             REFERENCES "THEATER_WORKER" ("ID") ENABLE
38.    );
39.
40.CREATE TABLE "STAGE_DIRECTOR"
41. (    "ID" NUMBER,
42.         PRIMARY KEY ("ID") ENABLE,
43.         FOREIGN KEY ("ID")
44.             REFERENCES "PRODUCER" ("ID") ENABLE
45.    );
46.
47. CREATE TABLE "ART_DIRECTOR"
48. (    "ID" NUMBER,
49.         PRIMARY KEY ("ID") ENABLE,
50.         FOREIGN KEY ("ID")
51.             REFERENCES "PRODUCER" ("ID") ENABLE
52.    );
53. CREATE TABLE "CONDUCTOR_DIRECTOR"
54. (    "ID" NUMBER,
55.         PRIMARY KEY ("ID") ENABLE,
56.         FOREIGN KEY ("ID")
57.             REFERENCES "PRODUCER" ("ID") ENABLE
58.    );
59.
60. CREATE TABLE "DIRECTOR"
61. (    "ID" NUMBER,
62.         PRIMARY KEY ("ID") ENABLE,
63.         FOREIGN KEY ("ID")
64.             REFERENCES "THEATER_WORKER" ("ID") ENABLE
65.    );
66.
67. CREATE TABLE "EMPLOYEE"
68. (    "ID" NUMBER NOT NULL ENABLE,
69.         PRIMARY KEY ("ID") ENABLE,
70.         FOREIGN KEY ("ID")
71.             REFERENCES "THEATER_WORKER" ("ID") ENABLE
72.    );
73.
74.CREATE TABLE "SUBSCRIPTION"
75. (    "ID" NUMBER,
76.         PRIMARY KEY ("ID") ENABLE
77.    );
78.
79.CREATE TABLE "AGE_CATEGORY"
80. (    "ID" NUMBER,
81.     "NAME" VARCHAR2(20),
82.         PRIMARY KEY ("ID") ENABLE
```

```
83.  );
84.
85.CREATE TABLE "AUTHOR"
86.  (  "ID" NUMBER,
87.      "COUNTRY" VARCHAR2(20),
88.      "CENTURY" NUMBER,
89.      "NAME" VARCHAR2(50),
90.      PRIMARY KEY ("ID") ENABLE
91.  );
92.
93.CREATE TABLE "AUTHOR_SUBSCRIPTION"
94.  (  "ID" NUMBER,
95.      "AUTHOR_ID" NUMBER,
96.      "PRICE" NUMBER,
97.      PRIMARY KEY ("ID") ENABLE,
98.      FOREIGN KEY ("AUTHOR_ID")
99.          REFERENCES "AUTHOR" ("ID") ENABLE,
100.     FOREIGN KEY ("ID")
101.        REFERENCES "SUBSCRIPTION" ("ID") ENABLE
102.  );
103.
104. CREATE TABLE "GENRE"
105.  (  "ID" NUMBER,
106.      "NAME" VARCHAR2(20),
107.      PRIMARY KEY ("ID") ENABLE
108.  );
109.
110. CREATE TABLE "GENRE_SUBSCRIPTION"
111.  (  "ID" NUMBER,
112.      "GENRE_ID" NUMBER,
113.      "PRICE" NUMBER,
114.      PRIMARY KEY ("ID") ENABLE,
115.      FOREIGN KEY ("ID")
116.          REFERENCES "SUBSCRIPTION" ("ID") ENABLE,
117.      FOREIGN KEY ("GENRE_ID")
118.        REFERENCES "GENRE" ("ID") ENABLE
119.  );
120.
121. CREATE TABLE "SPECTACLE"
122.  (  "ID" NUMBER,
123.      "ART_DIRECTOR_ID" NUMBER,
124.      "CONDUCTOR_DIRECTOR_ID" NUMBER,
125.      "STAGE_DIRECTOR_ID" NUMBER,
126.      "AUTHOR_ID" NUMBER,
127.      "MAIN_ACTOR_ID" NUMBER,
128.      "NAME" VARCHAR2(50),
129.      "GENRE_ID" NUMBER,
130.      "AGE_CATEGORY_ID" NUMBER,
131.      PRIMARY KEY ("ID") ENABLE,
132.      FOREIGN KEY ("ART_DIRECTOR_ID")
133.          REFERENCES "ART_DIRECTOR" ("ID") ENABLE,
134.      FOREIGN KEY ("MAIN_ACTOR_ID")
135.          REFERENCES "ACTOR" ("ID") ENABLE,
136.      FOREIGN KEY ("CONDUCTOR_DIRECTOR_ID")
137.          REFERENCES "CONDUCTOR_DIRECTOR" ("ID") ENABLE,
```

```
138.      FOREIGN KEY ("STAGE_DIRECTOR_ID")
139.          REFERENCES "STAGE_DIRECTOR" ("ID") ENABLE,
140.      FOREIGN KEY ("AUTHOR_ID")
141.          REFERENCES "AUTHOR" ("ID") ENABLE,
142.      FOREIGN KEY ("GENRE_ID")
143.          REFERENCES "GENRE" ("ID") ENABLE,
144.      FOREIGN KEY ("AGE_CATEGORY_ID")
145.          REFERENCES "AGE_CATEGORY" ("ID") ENABLE
146.      );
147.
148. CREATE TABLE "REPERTOIRE"
149. (
150.     "ID" NUMBER,
151.     "SPECTACLE_ID" NUMBER,
152.     "DATE_TIME" DATE,
153.     "PREMIERE_DATE" DATE,
154.     PRIMARY KEY ("ID") ENABLE,
155.     FOREIGN KEY ("SPECTACLE_ID")
156.         REFERENCES "SPECTACLE" ("ID") ENABLE
157.     );
158. CREATE TABLE "TICKET"
159. (
160.     "ID" NUMBER,
161.     "REPERTOIRE_NUMBER" NUMBER,
162.     "N_SITE" NUMBER,
163.     "N_ROW" NUMBER,
164.     "PRICE" NUMBER,
165.     PRIMARY KEY ("ID") ENABLE,
166.     FOREIGN KEY ("REPERTOIRE_NUMBER")
167.         REFERENCES "REPERTOIRE" ("ID") ENABLE
168.     );
169. CREATE TABLE "CASH_REGISTER"
170. (
171.     "PURCHASE_ID" NUMBER,
172.     "TICKET_ID" NUMBER,
173.     "SUBSCRIPTION_ID" NUMBER,
174.     PRIMARY KEY ("PURCHASE_ID") ENABLE,
175.     FOREIGN KEY ("TICKET_ID")
176.         REFERENCES "TICKET" ("ID") ENABLE,
177.     FOREIGN KEY ("SUBSCRIPTION_ID")
178.         REFERENCES "SUBSCRIPTION" ("ID") ENABLE
179.     );
180.
181. CREATE SEQUENCE sq_a_SUBSCRIPTION
182. START WITH 2
183. INCREMENT BY 2
184. NOMAXVALUE;
185.
186. CREATE SEQUENCE sq_g_SUBSCRIPTION
187. START WITH 1
188. INCREMENT BY 2
189. NOMAXVALUE;
190.
191. CREATE SEQUENCE sq_THEATER_WORKER
192. START WITH 1
```

```
193. INCREMENT BY 1
194. NOMAXVALUE;
195.
196. CREATE SEQUENCE sq_SPECTACLE
197. START WITH 1
198. INCREMENT BY 1
199. NOMAXVALUE;
200.
201. CREATE SEQUENCE sq_REPERTOIRE
202. START WITH 1
203. INCREMENT BY 1
204. NOMAXVALUE;
205.
206. CREATE SEQUENCE sq AGE CATEGORY
207. START WITH 1
208. INCREMENT BY 1
209. NOMAXVALUE;
210.
211. CREATE SEQUENCE sq TICKET
212. START WITH 1
213. INCREMENT BY 1
214. NOMAXVALUE;
215.
216. CREATE SEQUENCE sq_genre
217. START WITH 1
218. INCREMENT BY 1
219. NOMAXVALUE;
220.
221. CREATE SEQUENCE sq_author
222. START WITH 1
223. INCREMENT BY 1
224. NOMAXVALUE;
225.
226. CREATE SEQUENCE sq_cash
227. START WITH 1
228. INCREMENT BY 1
229. NOMAXVALUE;
230.
231. INSERT INTO
    THEATER_WORKER("LAST_NAME", "FIRST_NAME", "EMPLOYMENT_DAY", "GENDER", "BIRTHDAY",
    "SALARY", "CHILDRENS", "RANK") VALUES ('Bogdanov', 'Stanislav',
    to_date('01.05.2007', 'dd/mm/yyyy'), 'Male', to_date('07.09.1985',
    'dd/mm/yyyy'), 30000, 0, 'national_artist');
232. INSERT INTO
    THEATER_WORKER("LAST_NAME", "FIRST_NAME", "EMPLOYMENT_DAY", "GENDER", "BIRTHDAY",
    "SALARY", "CHILDRENS", "RANK") VALUES ('Pechkin', 'Petr', to_date('10.04.2006',
    'dd/mm/yyyy'), 'Male', to_date('04.11.1961', 'dd/mm/yyyy'), 51000, 3,
    'student');
233. INSERT INTO
    THEATER_WORKER("LAST_NAME", "FIRST_NAME", "EMPLOYMENT_DAY", "GENDER", "BIRTHDAY",
    "SALARY", "CHILDRENS", "RANK") VALUES ('Shubina', 'Anastasia',
    to_date('20.01.2011', 'dd/mm/yyyy'), 'Female', to_date('11.04.1995',
    'dd/mm/yyyy'), 41000, 1, 'guest_performer');
234. INSERT INTO
    THEATER_WORKER("LAST_NAME", "FIRST_NAME", "EMPLOYMENT_DAY", "GENDER", "BIRTHDAY", "
```

```

SALARY","CHILDRENS") VALUES ('Ostapchenko', 'Ivan', to_date('03.03.2013',
'dd/mm/yyyy'), 'Male', to_date('01.12.1986', 'dd/mm/yyyy'), 32000, 1);
235. INSERT INTO
    THEATER_WORKER("LAST_NAME", "FIRST_NAME", "EMPLOYMENT_DAY", "GENDER", "BIRTHDAY",
    "SALARY", "CHILDRENS") VALUES ('Bogdanova', 'Sofia', to_date('01.05.2007',
    'dd/mm/yyyy'), 'Female', to_date('07.09.1985', 'dd/mm/yyyy'), 30000, 0);
236. INSERT INTO
    THEATER_WORKER("LAST_NAME", "FIRST_NAME", "EMPLOYMENT_DAY", "GENDER", "BIRTHDAY",
    "SALARY", "CHILDRENS") VALUES ('Nagieva', 'Maria', to_date('23.03.2018',
    'dd/mm/yyyy'), 'Female', to_date('03.11.1999', 'dd/mm/yyyy'), 39000, 0);
237. INSERT INTO
    THEATER_WORKER("LAST_NAME", "FIRST_NAME", "EMPLOYMENT_DAY", "GENDER", "BIRTHDAY",
    "SALARY", "CHILDRENS") VALUES ('Taratutin', 'Vladislav', to_date('08.02.2020',
    'dd/mm/yyyy'), 'Male', to_date('07.06.1988', 'dd/mm/yyyy'), 45000, 1);
238. INSERT INTO
    THEATER_WORKER("LAST_NAME", "FIRST_NAME", "EMPLOYMENT_DAY", "GENDER", "BIRTHDAY",
    "SALARY", "CHILDRENS") VALUES ('Yatmanov', 'Oleg', to_date('12.04.2010',
    'dd/mm/yyyy'), 'Male', to_date('10.04.1995', 'dd/mm/yyyy'), 52000, 0);
239. INSERT INTO
    THEATER_WORKER("LAST_NAME", "FIRST_NAME", "EMPLOYMENT_DAY", "GENDER", "BIRTHDAY",
    "SALARY", "CHILDRENS") VALUES ('Beshneva', 'Olga', to_date('24.06.1985',
    'dd/mm/yyyy'), 'Female', to_date('11.10.1965', 'dd/mm/yyyy'), 40000, 1);
240. INSERT INTO
    THEATER_WORKER("LAST_NAME", "FIRST_NAME", "EMPLOYMENT_DAY", "GENDER", "BIRTHDAY",
    "SALARY", "CHILDRENS") VALUES ('Krasnoverova', 'Alisa', to_date('23.04.2009',
    'dd/mm/yyyy'), 'Female', to_date('07.09.1978', 'dd/mm/yyyy'), 55000, 2);
241. INSERT INTO
    THEATER_WORKER("LAST_NAME", "FIRST_NAME", "EMPLOYMENT_DAY", "GENDER", "BIRTHDAY",
    "SALARY", "CHILDRENS") VALUES ('Bessonov', 'Pavel', to_date('22.04.2012',
    'dd/mm/yyyy'), 'Male', to_date('03.03.1983', 'dd/mm/yyyy'), 47000, 1);
242. INSERT INTO
    THEATER_WORKER("LAST_NAME", "FIRST_NAME", "EMPLOYMENT_DAY", "GENDER", "BIRTHDAY",
    "SALARY", "CHILDRENS") VALUES ('Bessonova', 'Polina', to_date('24.08.2015',
    'dd/mm/yyyy'), 'Female', to_date('05.12.1985', 'dd/mm/yyyy'), 50000, 1);
243. INSERT INTO
    THEATER_WORKER("LAST_NAME", "FIRST_NAME", "EMPLOYMENT_DAY", "GENDER", "BIRTHDAY",
    "SALARY", "CHILDRENS") VALUES ('Dolgov', 'Ura', to_date('11.12.2011',
    'dd/mm/yyyy'), 'Male', to_date('01.04.1995', 'dd/mm/yyyy'), 37000, 0);
244. INSERT INTO
    THEATER_WORKER("LAST_NAME", "FIRST_NAME", "EMPLOYMENT_DAY", "GENDER", "BIRTHDAY",
    "SALARY", "CHILDRENS") VALUES ('Sazin', 'Stepan', to_date('07.02.2017',
    'dd/mm/yyyy'), 'Male', to_date('31.05.1994', 'dd/mm/yyyy'), 41000, 0);
245. INSERT INTO
    THEATER_WORKER("LAST_NAME", "FIRST_NAME", "EMPLOYMENT_DAY", "GENDER", "BIRTHDAY",
    "SALARY", "CHILDRENS") VALUES ('Rutashin', 'Oleg', to_date('24.02.2014',
    'dd/mm/yyyy'), 'Male', to_date('17.10.1991', 'dd/mm/yyyy'), 38000, 1);
246. INSERT INTO
    THEATER_WORKER("LAST_NAME", "FIRST_NAME", "EMPLOYMENT_DAY", "GENDER", "BIRTHDAY",
    "SALARY", "CHILDRENS") VALUES ('Ostashina', 'Maryana', to_date('03.08.2020',
    'dd/mm/yyyy'), 'Female', to_date('27.07.2000', 'dd/mm/yyyy'), 39000, 0);
247. INSERT INTO
    THEATER_WORKER("LAST_NAME", "FIRST_NAME", "EMPLOYMENT_DAY", "GENDER", "BIRTHDAY",
    "SALARY", "CHILDRENS") VALUES ('Lisina', 'Tatyana', to_date('04.09.2013',
    'dd/mm/yyyy'), 'Female', to_date('17.06.1987', 'dd/mm/yyyy'), 34000, 0);
248.

```

```

249. INSERT INTO ACTOR VALUES (1, 197, 98, 'yes');
250. INSERT INTO ACTOR VALUES (2, 173, 60, 'no');
251. INSERT INTO ACTOR VALUES (3, 185, 71, 'yes');
252. INSERT INTO ACTOR VALUES (4, 179, 69, 'yes');
253.
254. INSERT INTO DIRECTOR VALUES (5);
255.
256. INSERT INTO ART_DIRECTOR VALUES (6);
257. INSERT INTO CONDUCTOR_DIRECTOR VALUES (7);
258. INSERT INTO CONDUCTOR_DIRECTOR VALUES (8);
259. INSERT INTO STAGE_DIRECTOR VALUES (9);
260.
261. INSERT INTO MUSICIAN VALUES (10, 'ORGAN');
262. INSERT INTO MUSICIAN VALUES (11, 'VIOLIN');
263. INSERT INTO MUSICIAN VALUES (12, 'TRUMPET');
264.
265. INSERT INTO AGE_CATEGORY ("NAME") VALUES ('0+');
266. INSERT INTO AGE_CATEGORY ("NAME") VALUES ('6+');
267. INSERT INTO AGE_CATEGORY ("NAME") VALUES ('12+');
268. INSERT INTO AGE_CATEGORY ("NAME") VALUES ('16+');
269. INSERT INTO AGE_CATEGORY ("NAME") VALUES ('18+');
270.
271. INSERT INTO AUTHOR ("COUNTRY", "CENTURY", "NAME") VALUES ('Russia', '19',
    'Roman Urnyi');
272. INSERT INTO AUTHOR ("COUNTRY", "CENTURY", "NAME") VALUES ('France', '18',
    'Belle Bellini');
273. INSERT INTO AUTHOR ("COUNTRY", "CENTURY", "NAME") VALUES ('Germany', '17',
    'Brish Hatzmul');
274.
275. INSERT INTO GENRE ("NAME") VALUES ('Dramatic');
276. INSERT INTO GENRE ("NAME") VALUES ('Comedy');
277. INSERT INTO GENRE ("NAME") VALUES ('Tragedy');
278. INSERT INTO GENRE ("NAME") VALUES ('Musical');
279.
280. INSERT INTO SPECTACLE
    ("ART_DIRECTOR_ID", "CONDUCTOR_DIRECTOR_ID", "STAGE_DIRECTOR_ID", "AUTHOR_ID", "NA
    ME", "GENRE_ID", "AGE_CATEGORY_ID", "MAIN_ACTOR_ID")
281. VALUES (6, 7, 9, 1, 'ONE WINTER', 1, 3, 1);
282. INSERT INTO SPECTACLE
    ("ART_DIRECTOR_ID", "CONDUCTOR_DIRECTOR_ID", "STAGE_DIRECTOR_ID", "AUTHOR_ID", "NA
    ME", "GENRE_ID", "AGE_CATEGORY_ID", "MAIN_ACTOR_ID")
283. VALUES (6, 8, 9, 2, 'FLORENCE', 3, 4, 2);
284.
285. INSERT INTO REPERTOIRE ("SPECTACLE_ID", "DATE_TIME", "PREMIERE_DATE")
286. VALUES (1, to_date('10.03.2022 07:00', 'dd/mm/yyyy hh24:mi'),
    to_date('03.03.2012', 'dd/mm/yyyy'));
287.
288. CREATE OR REPLACE TRIGGER tr_author before INSERT ON author FOR each row
289.             BEGIN
290.                 SELECT sq_author.NEXTVAL
291.                     INTO :new.id
292.                     FROM dual;
293.             END;
294. CREATE OR REPLACE TRIGGER tr_A_PRODUCER after INSERTS
295.             ON ART_DIRECTOR

```

```

296.           for each row
297.           BEGIN
298.               INSERT INTO PRODUCER(id) VALUES (:NEW.id);
299.           END;
300. CREATE OR REPLACE TRIGGER tr_S_PRODUCER after INSERTs
301.             ON STAGE_DIRECTOR
302.             for each row
303.             BEGIN
304.                 INSERT INTO PRODUCER(id) VALUES (:NEW.id);
305.             END;
306. CREATE OR REPLACE TRIGGER tr_C_PRODUCER after INSERTs
307.             ON CONDUCTOR_DIRECTOR
308.             for each row
309.             BEGIN
310.                 INSERT INTO PRODUCER(id) VALUES (:NEW.id);
311.             END;
312. CREATE OR REPLACE TRIGGER tr_genre before INSERT ON genre FOR each row
313.             BEGIN
314.                 SELECT sq_genre.NEXTVAL
315.                   INTO :new.id
316.                   FROM dual;
317.             END;
318. CREATE OR REPLACE TRIGGER tr_TICKET before INSERT ON TICKET FOR each row
319.             BEGIN
320.                 SELECT sq_TICKET.NEXTVAL
321.                   INTO :new.id
322.                   FROM dual;
323.             END;
324. CREATE OR REPLACE TRIGGER tr AGE_CATEGORY before INSERT ON AGE_CATEGORY FOR
  each row
325.             BEGIN
326.                 SELECT sq AGE_CATEGORY.NEXTVAL
327.                   INTO :new.id
328.                   FROM dual;
329.             END;
330. CREATE OR REPLACE TRIGGER tr_REPERTOIRE before INSERT ON REPERTOIRE FOR
  each row
331.             BEGIN
332.                 SELECT sq_REPERTOIRE.NEXTVAL
333.                   INTO :new.id
334.                   FROM dual;
335.             END;
336. CREATE OR REPLACE TRIGGER tr_SPECTACLE before INSERT ON SPECTACLE FOR each
  row
337.             BEGIN
338.                 SELECT sq_SPECTACLE.NEXTVAL
339.                   INTO :new.id
340.                   FROM dual;
341.             END;
342. CREATE OR REPLACE TRIGGER tr_THEATER_WORKER before INSERT ON THEATER_WORKER
  FOR each row
343.             BEGIN
344.                 SELECT sq_THEATER_WORKER.NEXTVAL
345.                   INTO :new.id
346.                   FROM dual;

```

```

347.                                END;
348. CREATE OR REPLACE TRIGGER tr_id_a_SUBSCRIPTION before INSERT ON
    author_SUBSCRIPTION FOR each row
349.                                BEGIN
350.                                SELECT sq_a_SUBSCRIPTION.NEXTVAL
351.                                INTO :new.id
352.                                FROM dual;
353.                                END;
354. CREATE OR REPLACE TRIGGER tr_s_a_SUBSCRIPTION after INSERT
355.                                ON author_SUBSCRIPTION
356.                                for each row
357.                                BEGIN
358.                                INSERT INTO SUBSCRIPTION(id) VALUES (:NEW.id);
359.                                END;
360. CREATE OR REPLACE TRIGGER tr_id_g_SUBSCRIPTION before INSERT ON
    GENRE_SUBSCRIPTION FOR each row
361.                                BEGIN
362.                                SELECT sq_g_SUBSCRIPTION.NEXTVAL
363.                                INTO :new.id
364.                                FROM dual;
365.                                END;
366. CREATE OR REPLACE TRIGGER tr_s_g_SUBSCRIPTION after INSERT
367.                                ON GENRE_SUBSCRIPTION
368.                                for each row
369.                                BEGIN
370.                                INSERT INTO SUBSCRIPTION(id) VALUES (:NEW.id);
371.                                END;
372. CREATE OR REPLACE TRIGGER tr_CASH before INSERT ON CASH_REGISTER FOR each
    row
373.                                BEGIN
374.                                SELECT sq_CASH.NEXTVAL
375.                                INTO :new.PURCHASE_id
376.                                FROM dual;
377.                                END;
378.
379. create role c##DIRECTOR;
380. create role c##visiter;
381. create role c##admin;
382.
383. grant select on REPERTOIRE to c##visiter;
384. grant insert TICKET to c##visiter;
385. grant insert SUBSCRIPTION to c##visiter;
386. grant insert GENRE_SUBSCRIPTION to c##visiter;
387. grant insert AUTHOR_SUBSCRIPTION to c##visiter;
388. grant insert CASH_REGISTER to c##visiter;
389.
390. grant select on REPERTOIRE to c##director;
391. grant insert, delete, update on THEATER_WORKER to c##director;
392. grant insert, delete, update on ACTOR to c##director;
393. grant insert, delete, update on ART_DIRECTOR to c##director;
394. grant insert, delete, update on PRODUCER to c##director;
395. grant insert, delete, update on STAGE_DIRECTOR to c##director;
396. grant insert, delete, update on CONDUCTOR_DIRECTOR to c##director;
397. grant insert, delete, update on MUSICIAN to c##director;
398. grant insert, delete, update on SPECTACLE to c##director;

```

```

399. grant select on REPERTOIRE to c##director;
400. grant select on GENRE to c##director;
401. grant select on AGE_CATEGORY to c##director;
402. grant select on AUTHOR to c##director;
403.
404. grant insert, delete, update on REPERTOIRE to c##admin;
405. grant select on CASH_REGISTER to c##admin;
406. grant select on TICKET to c##admin;
407. grant select on SUBSCRIPTION to c##admin;
408. grant select on AUTHOR_SUBSCRIPTION to c##admin;
409. grant select on GENRE_SUBSCRIPTION to c##admin;
410. grant select on SPECTACLE to c##admin;
411.
412.
413. grant insert, delete, update on SPECTACLE to c##director;
414. create role c##adminDB;
415. grant c##director to c##admin;
416. grant insert, delete, update on REPERTOIRE to c##adminDB;
417. grant insert, delete, update on GENRE to c##adminDB;
418. grant insert, delete, update on AGE_CATEGORY to c##adminDB;
419. grant insert, delete, update on AUTHOR to c##adminDB;
420. commit;

```

## Глава 4. Тестирование и эксплуатация

Тестирование выполнялось следующим образом. Был предоставлен некоторый начальный набор тестовых данных. Эти данные были добавлены в СУБД с помощью скрипта. Далее, с помощью приложения выполнялись манипуляции с этими данными: удаление, изменение (где это доступно), а также добавление новых данных. Особое внимание было уделено проверке скриптов с ограничениями целостности. В результате финального тестирования всех возможностей приложения ошибок выявлено не было.

## Заключение

Во время анализа информационной системы были успешно выделены основные сущности и отношения между ними. Также, успешно были определены требования к обеспечению целостности данных и основные роли пользователей приложения.

Во время проектирования информационной системы была разработана архитектура приложения и ее основные части. Были разработаны проектные решения логического уровня и алгоритмы обеспечения целостности данных. Были раскрыты общие принципы построения интерфейса в соответствии с ролями пользователей и их прецедентами.

Во время реализации информационной системы были успешно реализованы все работы по программированию. Также, были реализованы в полной мере SQL скрипты для построения схемы данных и тестового набора данных. Был успешно реализован пользовательский интерфейс и обработка ошибок и исключений.

Во время тестирования были получены результаты о полном и корректном функционировании приложения и базы данных.