МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет

о научно-исследовательской работе

РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИНФОРМАЦИОННОЙ СИСТЕМЫ ТЕАТРА

Выполнил: студент 3-го курса гр. 17208
Гафиятуллин А.Р

Новосибирск, 2020

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ.

Цель проектного задания – создание информационной системы театра.

Назначение: использование в театрах.

При анализе проектного задания были выделены следующие бизнес-процессы:

- Получение информации о работе театра:
  - о спектаклях;
  - о персонале.
- Контроль за постановками спектаклей;
- Утверждение репертуара;
- Принятие на работу новых служащих;
- Приглашение актёров и постановщиков;
- Утверждение гастролей
- Получение экономической статистики по работе театра;
- Продажа билетов и абонементов.

Основные группы пользователей системы:

- Посетители театра;
- Работники театра.

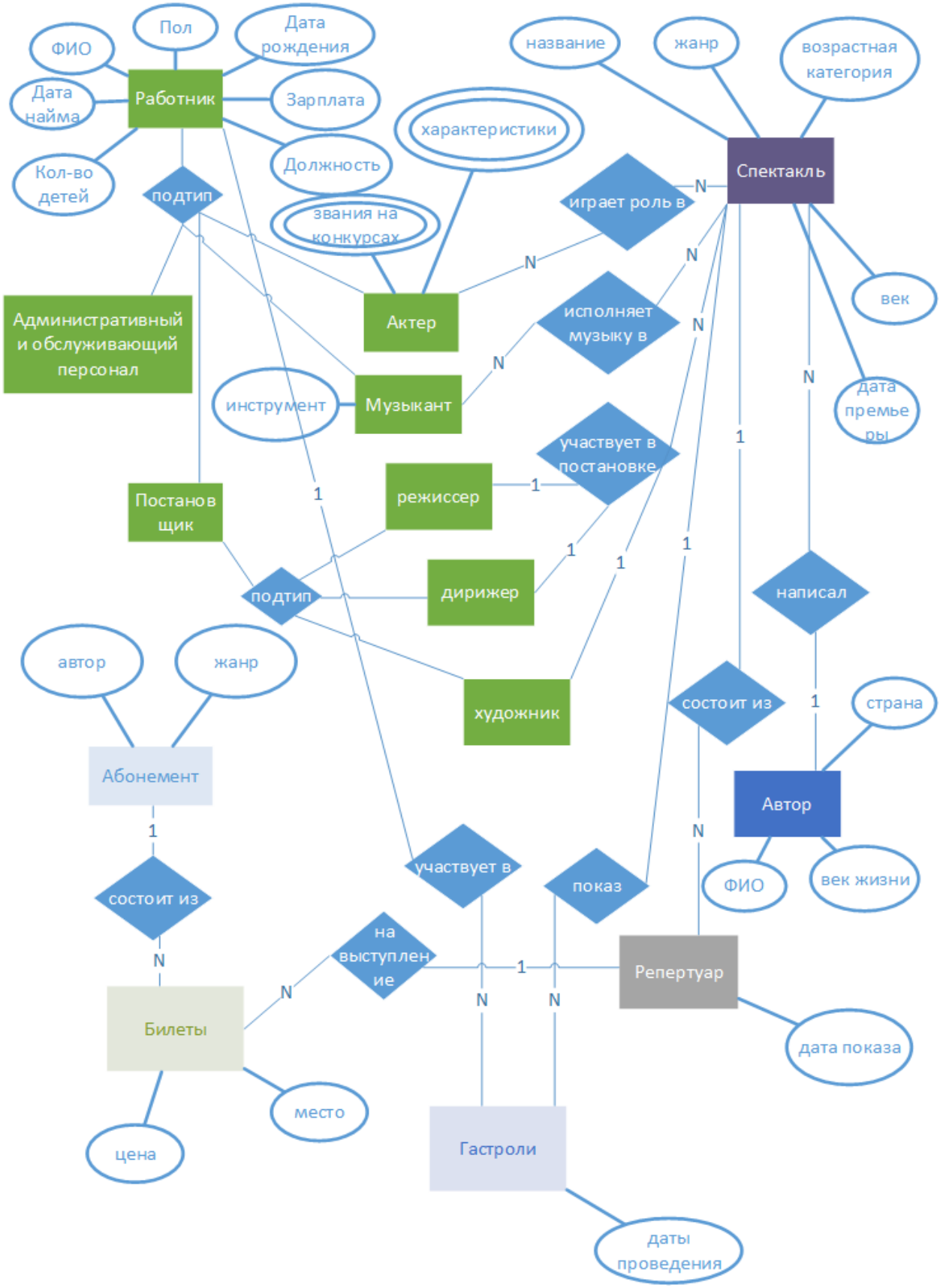
Задачи проектного задания:

- Анализ проекта;
- Проектирование системы;
- Реализация системы;
- Тестирование системы.

ГЛАВА 1. АНАЛИЗ ПРОЕКТА.

Основные сущности и отношения, определяемые (явно или неявно) проектным заданием приведены на ER-диаграмме ниже.

Группа сущностей с отношением супертип-подтип выделена зеленым цветом.


ФИО
Пол
Дата рождения
Дата найма
Работник
Зарплата
Кол-во детей
Должность
подтип
звания на конкурсах
характеристики
название
жанр
возрастная категория
Спектакль
играет роль в
N
N
Административный и обслуживающий персонал
Актер
исполняет музыку в
N
N
век
инструмент
Музыкант
N
N
дата премьеры
участвует в постановке
1
1
Постановщик
1
режиссер
1
1
1
подтип
дирижер
1
написал
автор
жанр
художник
состоит из
1
страна
Абонемент
Автор
1
N
участвует в
показ
ФИО
век жизни
состоит из
N
на выступление
1
Репертуар
N
N
N
Билеты
дата показа
место
цена
Гастроли
даты проведения

Требования к обеспечению целостности данных:

1. согласованность дат и веков, т. е., например дата найма сотрудника не может быть раньше даты его рождения, дата показа спектакля не может быть раньше даты премьеры и т. д.;
2. запрет на удаление информации, которая все еще может быть полезна в запросах информационной системы, например невозможно удаление актера, постановщика или музыканта, который задействован в спектакле;
3. выбор работников с подходящей должностью(профессией) на различные позиции при постановке спектакля;
4. запрет назначения показа спектакля, для которого еще полностью не сформирован актерский состав;
5. запрет назначения пересекающихся по времени выступлений;
6. контроль за ошибками назначения актера на несколько ролей в одном и том же спектакле;
7. контроль за модификацией информации о спектакле, который уже в репертуаре театра;
8. назначение на роли в спектакле только тех актеров, которые подходят под требования данной роли;
9. контроль за эмиссией новых билетов, например, не должно быть двух билетов на одно и то же место на одно и то же выступление;
10. контроль за формированием абонементов: должны быть согласованы жанр или автор абонемента и входящих в него билетов, один билет не может попасть в два разных абонемента;
11. контроль за продажей билетов: правильные дата и время продажи;
12. при продаже абонемента должны быть проданы билеты, входящие в него;
13. контроль назначения гастролей: сотрудники должны принимать участие в спектакле, с которым они едут на гастроли, причем у них не должно быть назначено пересекающихся гастролей;
14. автоматическая инкрементация ключа во всех таблицах с первичными ключами.

Основные пути обеспечения целостности:

- на уровне базы данных:
    - триггеры вставки, обновления и удаления;

    - хранимые процедуры;
    - каскадное удаление данных.
- на уровне клиентского приложения:
    - информация для ввода выдается выпадающими списками и только та, которая необходима;
    - отсутствуют элементы интерфейса, позволяющие пользователю как-либо навредить целостности базы данных.

Роли для разрабатываемого приложения:

При анализе проектного задания были выделены следующие роли пользователей и основные сценарии использования:

Роль:	Вариант использования:	Номера запросов из задания:	<<uses>> варианты использования роли:	<<extends>> варианты использования роли:
Пользователь	Все базовые операции БД по получению информации о работе театра	2, 3, 4, 5, 7, 8, 9, 10	-	-
Директор	Контроль за постановками спектаклей, утверждение репертуара, принятие на работу новых служащих, приглашение актёров и постановщиков, утверждение гастролей	1, 6	Пользователь	Администрация
Администрация	Получение экономической статистики по работе театра	11, 12	Пользователь	-
Кассир	Продажа билетов и абонементов	13	Пользователь	-

ГЛАВА 2. ПРОЕКТИРОВАНИЕ СИСТЕМЫ.

Архитектура приложения: клиент-сервер.

a. в качестве сервера выступает машина с установленной БД Oracle;
b. в качестве клиента выступает любая машина, поддерживающая виртуальную машину Java и имеющая соединение с сетью Интернет.
c. Алгоритм взаимодействия клиента и сервера:
a. клиентское приложение связывается по известному IP-адресу с сервером посредством JDBC.
b. клиентское приложение проводит аутентификацию пользователя;
c. клиентское приложение предоставляет пользователю формы, необходимые его роли;
d. клиентское приложение совершает вызовы хранимых процедур БД посредством JDBC в зависимости от задач пользователя;
e. клиентское приложение завешает сеанс работы с сервером.

Основные таблицы и группы таблиц:

- Таблицы с характеристиками и свойствами:
  - Characteristic (характеристики актеров);
  - Gender (пол);
  - Education (уровни образования);
  - Job_types (должности(профессии));
  - Age_category (возрастные категории);
  - Genre (жанры спектаклей);
  - Country (страны);
  - Rank (звания);
  - Competition (конкурсы, на которых можно получить звания);
  - Musical_instruments (музыкальные инструменты).
- Основные таблицы, реализующие сущности:
  - Employee (работники);
  - Show (спектакли);
  - Role (роли спектаклей);
  - Repertoire (репертуар);
  - Author (авторы);
  - Tour (гастроли);
  - Ticket (билеты);
  - Subscription (Абонементы).
- Таблицы, реализующие отношения между сущностями и атрибуты этих сущностей:

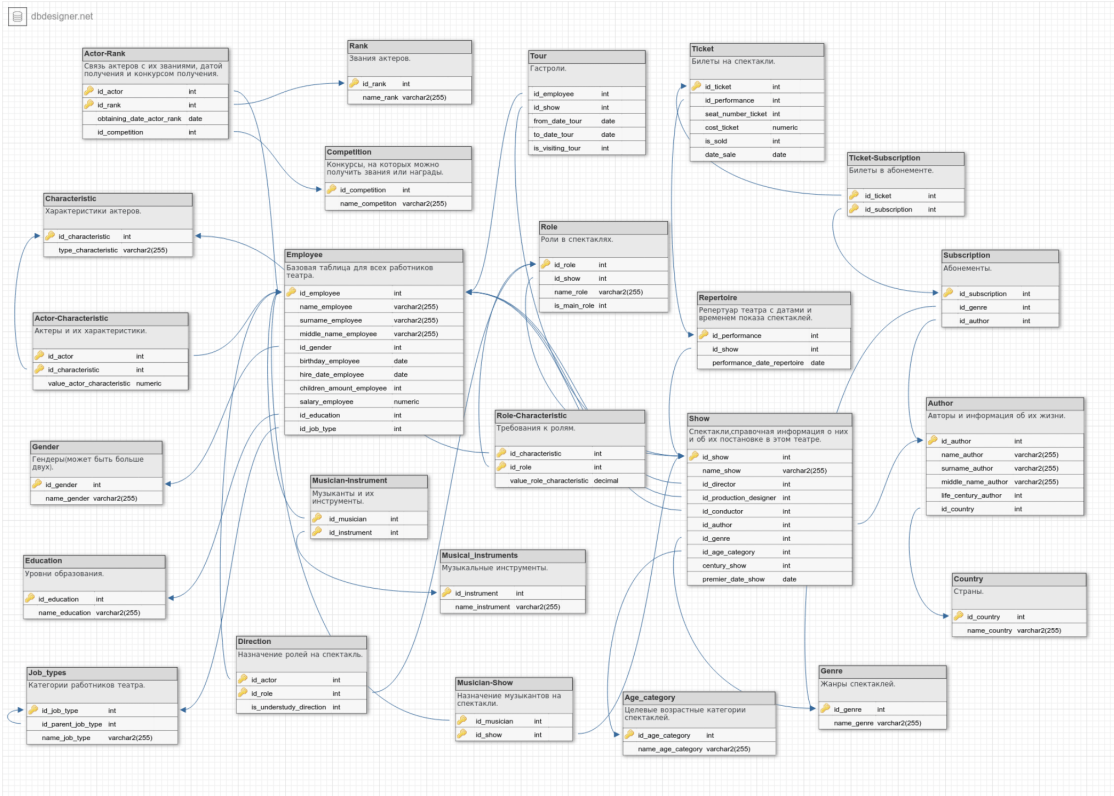
- Actor-Rank (актеры и их звания);
- Actor-Characteristic (характеристики актеров);
- Direction (назначение актеров на роли);
- Musician-Show (назначение музыкантов на спектакли);
- Musician-instrument (музыканты и их инструменты);
- Role-Characteristic (требования к актеру для назначения на роль);
- Ticket- Subscription (состав абонемента из билетов).

Способ представления супертипов и подтипов:

a. иерархия работников:
   i. супертип работника представлен таблицей Employee;
   ii. подтипы определяются полем профессии в таблице Employee из таблицы Job_types и набором триггеров, которые реагируют на подтипы в зависимости от значения в этом поле.
b. иерархия типов профессий:
   в таблице Job_types 3 поля: id_job_type – номер профессии, id_parent_job_type – номер родительской профессии и name_job_type – название профессии. Тип профессии высший в иерархии имеет в поле id_parent_job_type значение Null, а его подтипы в поле id_parent_job_type имеют id_job_type этой профессии.

Диаграмма схемы БД:

pdf со схемой БД: theatre-dbdesigner.pdf



Алгоритмы обеспечения целостности данных:

№	Уровень	Алгоритм
1	Ядро СУБД	Соединение необходимых таблиц, получение дат, веков и проверка на противоречие этих дат, веков. При противоречии - генерация исключения.
2	Ядро СУБД	Соединение необходимых таблиц, подсчет количества записей, проверка этого количества на равенство нулю. При противоречии - генерация исключения.
3	Клиентское приложение	Элементы интерфейса дают возможность выбора только подходящих профессий.
4	Ядро СУБД	Получение курсора ролей для спектакля, итерация по этому курсору с поиском количества актеров, назначенных на роль. Если менее одного для обычной роли и менее двух для главной, то генерация исключения.
5	Ядро СУБД	Получение курсора выступлений для сотрудника и сравнение с датами нового выступления. При пересечении – генерация исключения.
6	Ядро СУБД	Получение количества актеров для данной роли, если оно равно 1 для обычной роли или больше 2 для главной или попытка назначения двух главных или двух дублеров актеров на главную роль, то генерация исключения.

7	Ядро СУБД	При попытке модификации назначенных актеров, музыкантов или постановщиков на спектакль, происходит поиск спектакля в числе уже показываемых. В случае наличия – генерация исключения.
8	Хранимые процедуры	В качестве кандидатов на роль выдаются только те актеры, чьи характеристики являются надмножеством характеристик, требуемых ролью.
9	Ядро СУБД	Проверка существования в таблице билета на указанное место на указанное выступление. В случае наличия – генерация исключения.
10	Ядро СУБД	Соединение таблиц с Билетами, Репертуаром и Спектаклями, получение их этой таблицы жанра и автора спектакля, сравнение с жанром иди автором абонемента. В случае несовпадения – генерация исключения.
11	Хранимые процедуры	При пометке билета в качестве проданного, происходит получение времени функциями СУБД.
12	Хранимые процедуры	Множественный вызов хранимой процедуры из пункта 11 на основе таблицы соотношения абонементов и билетов.
13	Ядро СУБД	Проверка того, что сотрудник является актером, музыкантом или постановщиком в спектакле. Получение курсора гастролей для сотрудника и сравнение с датами новых гастролей. При невыполнении хотя бы одного условия – генерация исключения.
14	Ядро СУБД	Создание последовательности и получение из нее очередного значения при вставке новой записи в таблицу.

Перечень форм:

Форма:	Описание:	Варианты использования:
Спектакли: информация	Интерфейс для получения информации спектаклях	Все базовые операции БД по получению информации о спектаклях (запросы № 2, 3, 5, 9)
Спектакли: редактирование	Интерфейс для добавления и редактирования информации о спектаклях	Контроль за постановками спектаклей, утверждение репертуара (запрос № 6)
Служащие	Интерфейс для работы с информацией о служащих	Принятие на работу новых служащих или их увольнение (запрос № 1)
Актеры: информация	Интерфейс для получения информации об актерах	Все базовые операции БД по получению информации об актерах (запросы № 7, 10)
Актеры: редактирование	Интерфейс для добавления и редактирования информации об актерах	Приглашение актёров
Гастроли	Интерфейс ля работы с информацией о гастролях	Получение информации о гастролях (запрос № 8), утверждение гастролей

Постановщики	Интерфейс для работы с информацией о постановщиках	Все базовые операции БД по получению информации о постановщиках (запрос № 8), приглашение постановщиков
Музыканты	Интерфейс для работы с информацией о музыкантах	Получение информации о музыкантах, приглашение музыкантов
Авторы	Интерфейс для работы с информацией об авторах	Получения информации об авторах (запрос № 4), добавление и редактирование информации об авторах, нужно для спектаклей
Билеты и абонементы: продажа	Интерфейс для поиска и продажи доступных билетов и абонементов, соответствующих требованиям покупателя	Продажа билетов и абонементов (запрос № 13)
Билеты и абонементы: добавление	Интерфейс для добавления новых билетов и абонементов	Добавление новых билетов и абонементов
Экономические показатели	Интерфейс для получения экономических показателей работы театра	Получение экономической статистики по работе театра (запросы № 11, 12)
Театр	Интерфейс для получения различной информации	Доступ к другим формам в зависимости от роли пользователя
Вспомогательная форма	Интерфейс для добавления вспомогательной информации в таблицу	Добавление новых званий актеров, характеристик работников, гендеров, уровней образования, категорий работников, конкурсов, возрастных категорий, жанров спектаклей, стран, музыкальных инструментов.
Логин	Интерфейс для входа в систему	Вход в систему с использованием учетной записи

Авторизация:

- Клиент обеспечивает ввод и передачу логина и пароля на серверную сторону. После чего получает роль входящего пользователя и отображает необходимые ему формы.
- СУБД получает запрос на проверку существования пары логин-пароль. В случае существования возвращает роль, а иначе происходит генерация исключения.

ГЛАВА 3. РЕАЛИЗАЦИЯ СИСТЕМЫ.

Общий объем работ по программированию:

- Клиентская часть:
  - 15 форм на языке программирования Java;
- Серверная часть на языках SQL и PL\SQL:
  - скрипт создания 26 таблиц в СУБД;
  - скрипт создания 29 триггеров в СУБД;
  - скрипт создания 45 хранимых процедур в СУБД;
  - скрипт удаления ограничений целостности, последовательностей, триггеров, таблиц.

Номенклатура SQL-скриптов для построения схемы данных с иллюстрированием фрагментами кода:

- скрипт theatre_oracle_create.sql создает схему БД с ограничениями целостности по внешним ключам и каскадными удалениями данных.
  Пример:

```
CREATE TABLE "Employee" (
        "id_employee" INT PRIMARY KEY,
        "name_employee" VARCHAR2(255) NOT NULL,
        "surname_employee" VARCHAR2(255),
        "middle_name_employee" VARCHAR2(255),
        "id_gender" INT NOT NULL,
        "birthday_employee" DATE NOT NULL,
        "hire_date_employee" DATE NOT NULL,
        "children_amount_employee" INT DEFAULT 0
CHECK("children_amount_employee" >= 0),
        "salary_employee" NUMERIC(*, 2) NOT NULL
CHECK("salary_employee" > 0),
        "id_education" INT NOT NULL,
        "id_job_type" INT NOT NULL);

CREATE sequence "EMPLOYEE_ID_EMPLOYEE_SEQ";
/
ALTER TABLE "Employee" ADD CONSTRAINT "Employee_fk0"
FOREIGN KEY ("id_gender") REFERENCES "Gender"("id_gender");
```

```
ALTER TABLE "Employee" ADD CONSTRAINT "Employee_fk1"
FOREIGN KEY ("id_education") REFERENCES
"Education"("id_education");
ALTER TABLE "Employee" ADD CONSTRAINT "Employee_fk2"
FOREIGN KEY ("id_job_type") REFERENCES
"Job_types"("id_job_type");
```

- скрипт theatre_oracle_drop.sql удаляет ограничения целостности, последовательности, триггеры, таблицы. Пример:

```
ALTER TABLE "Ticket-Subscription" DROP CONSTRAINT "Ticket-
Subscription_fk0";
ALTER TABLE "Ticket-Subscription" DROP CONSTRAINT "Ticket-
Subscription_fk1";

DROP TABLE "Rank";
DROP TABLE "Musical_instruments";
```

Номенклатура PL\SQL-скриптов для обеспечения целостности данных:

- скрипт theatre_oracle_triggers.sql создает триггеры, необходимые для ограничения целостности БД. Пример:

```
CREATE OR REPLACE trigger "REPERTOIRE-INSERT"
   before insert on "Repertoire"
   for each row
declare
   premier_date DATE;
   actors_count INT;
   cursor show_roles is
      select "id_role", "is_main_role"
      from "Role"
      where "id_show" = :NEW."id_show";
   performance_amount INT;

begin
   select "premier_date_show" into premier_date
   from "Show"
   where "id_show" = :NEW."id_show";
```

```
    if :NEW."performance_date_repertoire" < premier_date then
        raise_application_error(-20019, 'Дата показа не может быть раньше
даты премьеры!');
    end if;

    for role_record in show_roles
    loop
        select count(*) into actors_count
        from "Direction"
        where "id_role" = role_record."id_role";

        if role_record."is_main_role" = 0 and actors_count < 1 or
role_record."is_main_role" = 1 and actors_count < 2 then
            raise_application_error(-20020, 'Актерский состав для данного
спектакля сформирован не полностью!');
        end if;
    end loop;

    select count(*) into performance_amount
    from "Repertoire"
    where "performance_date_repertoire" =
:NEW."performance_date_repertoire";

    if performance_amount != 0 then
        raise_application_error(-20021, 'На это время уже назначен
спектакль!');
    end if;

    select "REPERTOIRE_ID_PERFORMANCE_SEQ".nextval into
:NEW."id_performance" from dual;
end;
```

- скрипт theatre_oracle_stored_procedures.sql заводит в БД хранимые процедуры и функции. Часть параметров может быть NULL, используется функция NVL, чтобы не учитывать этот параметр при запросе (сделать его всегда истинным). Пример:

```
CREATE OR REPLACE procedure show_info(
   from_date_show IN VARCHAR2,
   to_date_show IN VARCHAR2,
   first_time_show IN INT,
   id_show IN INT,
   status IN INT,
   from_century_show IN INT,
   to_century_show IN INT,
   id_conductor IN INT,
   id_production_designer IN INT,
   id_director IN INT,
   id_genre IN INT,
   id_age_category IN INT,
   id_author IN INT,
   id_country IN INT,
   show_cur OUT SYS_REFCURSOR)
is
begin
   open show_cur for
      select "id_show", "name_show" as "название",
         (select "name_employee" || ' ' || "surname_employee" || ' ' ||
"middle_name_employee"
         from "Employee" where "id_employee" = "id_director") as
"режиссер-постановщик",
      (select "name_employee" || ' ' || "surname_employee" || ' ' ||
"middle_name_employee"
         from "Employee" where "id_employee" = "id_conductor") as
"дирижер-постановщик",
      (select "name_employee" || ' ' || "surname_employee" || ' ' ||
"middle_name_employee"
         from "Employee" where "id_employee" = "id_production_designer")
as "художник-постановщик",
      ("name_author" || ' ' || "surname_author" || ' ' || "middle_name_author")
as "автор",
      "name_genre" as "жанр",
      "century_show" as "век",
      "premier_date_show" as "премьера"
```

```
    from (("Show" inner join "Author" using("id_author")) inner join
"Genre" using("id_genre"))
    where "id_show" in (select "id_show"
                from (("Repertoire" inner join "Show" using ("id_show"))
                    inner join "Author" using ("id_author"))
                where
                  ("performance_date_repertoire" <=
NVL(TO_DATE(to_date_show, 'yyyy/mm/dd'),
                    "performance_date_repertoire")
                  and "performance_date_repertoire" >=
NVL(TO_DATE(from_date_show, 'yyyy/mm/dd'),
                    "performance_date_repertoire")
                  and ((status = 0) or "performance_date_repertoire" <=
(select CURRENT_DATE from dual)))

                  and "id_show" = NVL(id_show, "id_show")
                  and "century_show" <= NVL(to_century_show,
"century_show")
                    and "century_show" >= NVL(from_century_show,
"century_show")
                  and "id_conductor" = NVL(id_conductor,
"id_conductor")
                  and "id_production_designer" =
NVL(id_production_designer, "id_production_designer")
                  and "id_director" = NVL(id_director, "id_director")
                  and "id_genre" = NVL(id_genre, "id_genre")
                  and "id_age_category" = NVL(id_age_category,
"id_age_category")
                  and "id_author" = NVL(id_author, "id_author")
                  and "id_country" = NVL(id_country, "id_country")
                group by "id_show"
                having first_time_show = 0 or (first_time_show != 0 and
count(*) = 1));
end;
/
```

Характеристики тестового набора данных:

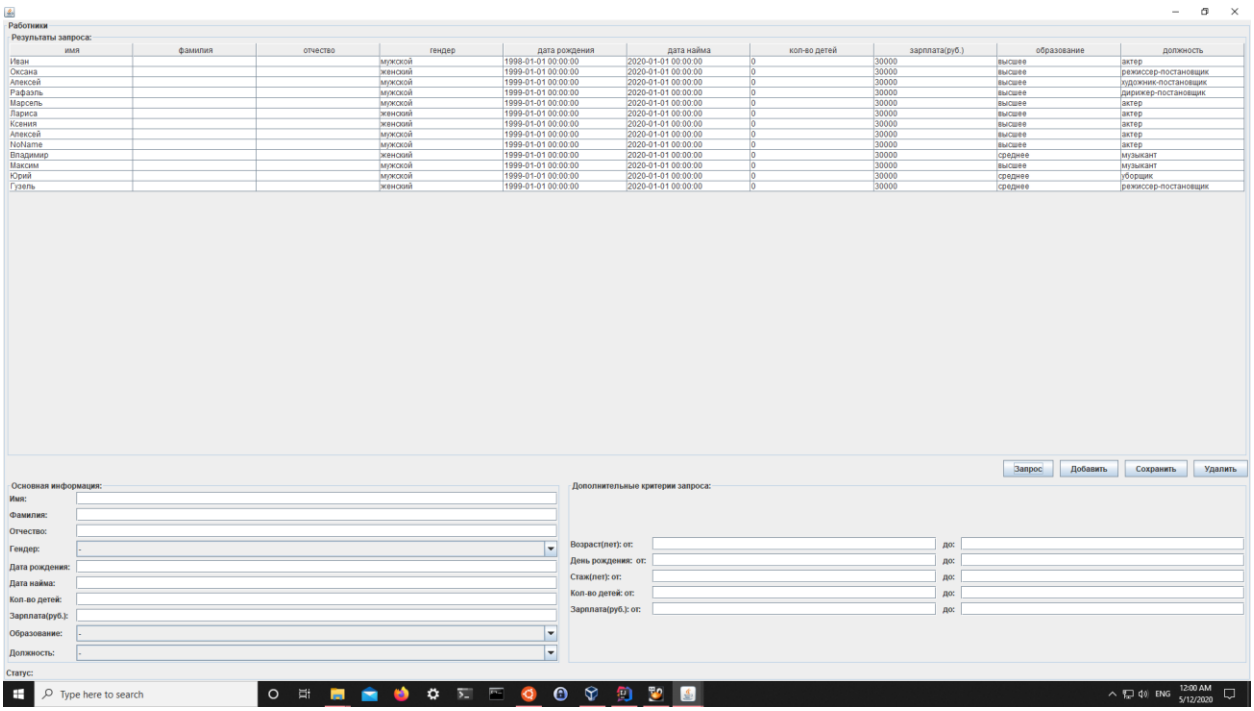
- тестируют правильность заполнения таблиц;
- тестируют правильность работы триггеров, ограничивающих целостность БД.

Пример из скрипт theatre_oracle_test.sql осуществляет тестовое заполнение таблиц:

```
INSERT INTO "Author" VALUES(0, 'Людвиг', 'ван Бетховен', '', 18, 1);
/* должно выполниться */

INSERT INTO "Author" VALUES(0, 'Людвиг ван', 'Бетховен', '', 18, 1);
/* должно выполниться */

INSERT INTO "Author" VALUES(0, 'NoName', '', '', 18, 1);
/* должно выполниться */

DELETE FROM "Author" WHERE "name_author" LIKE 'NoName';
/* должно выполниться */


DELETE FROM "Country" WHERE "name_country" LIKE 'Священная Римская империя';                    /* не должно выполниться */
```

Реализация пользовательского интерфейса системы:

- реализован на SWING с использованием встроенного в Intellij IDEA редактора форм.
- пример формы:



Обработка ошибок и исключений происходит на клиентской стороне. При получении ошибки отображается JOptionPane с описанием и кодом ошибки, полученной из БД. Пример:

```
try {
        String name = null;
        if(!nameTextField.getText().isEmpty()) {
            name = nameTextField.getText();
        }
        String surname = null;
        if(!surnameTextField.getText().isEmpty()) {
            surname = surnameTextField.getText();
        }
        String middle_name = null;
        if(!middleNameTextField.getText().isEmpty()) {
            middle_name = middleNameTextField.getText();
```

```
            }
            int country = countries.get(countryComboBox.getSelectedItem());
            String century_str = century.getText();
            if(name == null || name.isEmpty() || country == 0 || century_str == null
|| century_str.isEmpty()) {
                JOptionPane.showMessageDialog(mainPanel, "Не все поля
заполнены!",
                        "Ошибка добавления", JOptionPane.ERROR_MESSAGE);
            } else {
                addAuthor.setString(1, name);
                addAuthor.setString(2, surname);
                addAuthor.setString(3, middle_name);
                addAuthor.setInt(4, Integer.parseInt(century_str));
                addAuthor.setInt(5, country);
                addAuthor.execute();

                updateResultTable();
            }
        } catch (Exception exception) {
            JOptionPane.showMessageDialog(mainPanel,
exception.getMessage(),
                    "Ошибка добавления", JOptionPane.ERROR_MESSAGE);
            exception.printStackTrace();
        }
```

ГЛАВА 4. ТЕСТИРОВАНИЕ.

Критерии для тестового набора данных:

- должен тестировать правильность работы триггеров при ограничении целостности;
- должен тестировать правильность работы хранимых процедур и функций;
- должен быть автоматизированным для проверки правильности работы приложения при внесении изменений в него.

Порядок работ по тестированию приложения:

- проводится проверка правильности заполнения данными или удаления из БД данных с учетом триггеров, срабатывающих на нарушение целостности;
- проводится проверка правильности возврата хранимыми процедурами и функциями результатов на основе данных, которыми была заполнена БД на 1 этапе.

Результаты:

- Приложение протестировано и реализует правильную работу в соответствии с проектным заданием.

ЗАКЛЮЧЕНИЕ.

1. ***Проведен анализ проекта:***
    a. Описаны основные сущности и отношения, определяемые проектным заданием, выделены группы сущностей, определено наличие отношений супертип - подтип, приведена ER диаграмма;
    b. Выявлены требования к обеспечению целостности данных, и основных путях обеспечения их выполнения;
    c. Выявлены основные роли пользователей приложения и основные сценарии использования их взаимодействия с приложением. Приведена диаграмма прецедентов;
    d. Выполнено полно и качественно.
2. ***Проведено проектирование системы.***

a. Приведена общая архитектура приложения и ее основные части с описанием алгоритмов их взаимодействия;
b. Описаны проектные решения логического уровня, включая основные таблицы и их группы, способ представления в реляционной схеме супертипов и подтипов сущностей с обоснованием выбора, сложных моментов логического проектирования и неочевидных решений. Приведена диаграмма схемы БД;
c. Построены алгоритмы обеспечения целостности данных, определено разделение ответственности за обеспечение целостности между ядром СУБД, слоем хранимых процедур и алгоритмами клиентской части приложения;
d. Раскрыто общее строение интерфейса в соответствии с ролями пользователей и прецедентами. Приведена таблица роли - прецеденты - формы с перечнем наиболее принципиальных форм интерфейса с кратким описанием функциональности каждой;
e. Решены вопросы авторизации и ее разнесения на уровни СУБД и клиентской части приложения;
f. Выполнено полно и качественно.

3. ***Система реализована.***
    a. Описан общий объем работ по программированию (в соответствии с ранее описанной архитектурой;
    b. Описана номенклатура SQL скриптов для построения схемы данных с иллюстрированием фрагментами кода;
    c. Приведены характеристики тестового набора данных и скрипты SQL для ввода тестового набора данных в систему;
    d. Описана номенклатура PL/SQL скриптов, обеспечивающих различные аспекты целостности данных с иллюстрированием фрагментами кода;
    e. Описана реализация пользовательского интерфейса системы;
    f. Освещены вопросы обработки ошибок и исключений с иллюстрированием программным кодом;
    g. Выполнено полно и качественно.
4. ***Система протестирована.***
    a. Приведены сведения о критериях для тестового набора данных;
    b. Перечислен порядок работ по тестированию приложения;

c. Описаны результаты проведенного тестирования в связи с проектным заданием;
d. Выполнено полно и качественно.

Поставленная цель полностью достигнута.

ЛИТЕРАТУРА.

1. ER-модель, Нотация П. Чена
2. Фейерштейн С., Прибыл Б. Oracle PL/SQL. Для профессионалов. 6-е изд. — СПб.: Питер, 2015. — 1024 с.: ил. —(Серия «Бестселлеры O’Reilly»).
3. Ресурсы StackOverflow. – URL: https://stackoverflow.com/

ПРИЛОЖЕНИЕ 1. СКРИПТ СОЗДАНИЯ СХЕМЫ БД.

```
CREATE TABLE "Rank" (

    "id_rank" INT PRIMARY KEY,

    "name_rank" VARCHAR2(255) UNIQUE NOT NULL);



CREATE sequence "RANK_ID_RANK_SEQ";



CREATE trigger "BI_RANK_ID_RANK"

   before insert on "Rank"

   for each row

begin

   select "RANK_ID_RANK_SEQ".nextval into :NEW."id_rank" from dual;

end;

/
```

```
CREATE TABLE "Employee" (
        "id_employee" INT PRIMARY KEY,
        "name_employee" VARCHAR2(255) NOT NULL,
        "surname_employee" VARCHAR2(255),
        "middle_name_employee" VARCHAR2(255),
        "id_gender" INT NOT NULL,
        "birthday_employee" DATE NOT NULL,
        "hire_date_employee" DATE NOT NULL,
        "children_amount_employee" INT DEFAULT 0
CHECK("children_amount_employee" >= 0),
        "salary_employee" NUMERIC(*, 2) NOT NULL
CHECK("salary_employee" > 0),
        "id_education" INT NOT NULL,
        "id_job_type" INT NOT NULL);

CREATE sequence "EMPLOYEE_ID_EMPLOYEE_SEQ";
/

CREATE TABLE "Musician-Show" (
    "id_musician" INT NOT NULL,
```

```
    "id_show" INT NOT NULL,
    constraint MUSICIAN_SHOW_PK PRIMARY KEY
("id_musician","id_show"));
/

CREATE TABLE "Gender" (
        "id_gender" INT PRIMARY KEY,
        "name_gender" VARCHAR2(255) UNIQUE NOT NULL);

CREATE sequence "GENDER_ID_GENDER_SEQ";

CREATE trigger "BI_GENDER_ID_GENDER"
    before insert on "Gender"
    for each row
begin
    select "GENDER_ID_GENDER_SEQ".nextval into :NEW."id_gender" from
dual;
end;
/

CREATE TABLE "Actor-Characteristic" (
```

```
        "id_actor" INT PRIMARY KEY,
        "id_characteristic" INT NOT NULL,
        "value_actor_characteristic" NUMERIC(*, 2));
/

CREATE TABLE "Competition" (
        "id_competition" INT PRIMARY KEY,
        "name_competition" VARCHAR2(255) UNIQUE NOT NULL);

CREATE sequence "COMPETITION_ID_COMPETITON_SEQ";

CREATE trigger "BI_COMPETITION_ID_COMPETITON"
   before insert on "Competition"
   for each row
begin
   select "COMPETITION_ID_COMPETITON_SEQ".nextval into
:NEW."id_competition" from dual;
end;
/
```

```
CREATE TABLE "Actor-Rank" (
        "id_actor" INT NOT NULL,
        "id_rank" INT NOT NULL,
        "obtaining_date_actor_rank" DATE NOT NULL,
        "id_competition" INT,
        constraint ACTOR_RANK_PK PRIMARY KEY ("id_actor","id_rank"));
/

CREATE TABLE "Characteristic" (
        "id_characteristic" INT PRIMARY KEY,
        "type_characteristic" VARCHAR2(255) UNIQUE NOT NULL);

CREATE sequence "CHARACTERISTIC_ID_CHARACTERISTIC_SEQ";

CREATE trigger "BI_CHARACTERISTIC_ID_CHARACTERISTIC"
   before insert on "Characteristic"
   for each row
begin
   select "CHARACTERISTIC_ID_CHARACTERISTIC_SEQ".nextval into
:NEW."id_characteristic" from dual;
```

```
end;
/

CREATE TABLE "Education" (
        "id_education" INT PRIMARY KEY,
        "name_education" VARCHAR2(255) UNIQUE NOT NULL);

CREATE sequence "EDUCATION_ID_EDUCATION_SEQ";

CREATE trigger "BI_EDUCATION_ID_EDUCATION"
   before insert on "Education"
   for each row
begin
   select "EDUCATION_ID_EDUCATION_SEQ".nextval into
:NEW."id_education" from dual;
end;
/

CREATE TABLE "Show" (
        "id_show" INT PRIMARY KEY,
```

```
	"name_show" VARCHAR2(255) UNIQUE NOT NULL,
	"id_director" INT NOT NULL,
	"id_production_designer" INT NOT NULL,
	"id_conductor" INT NOT NULL,
	"id_author" INT NOT NULL,
	"id_genre" INT NOT NULL,
	"id_age_category" INT NOT NULL,
	"century_show" INT NOT NULL,
	"premier_date_show" DATE NOT NULL);

CREATE sequence "SHOW_ID_SHOW_SEQ";
/

CREATE TABLE "Author" (
	"id_author" INT PRIMARY KEY,
	"name_author" VARCHAR2(255) NOT NULL,
	"surname_author" VARCHAR2(255),
	"middle_name_author" VARCHAR2(255),
	"life_century_author" INT NOT NULL,
	"id_country" INT NOT NULL);
```

```
CREATE sequence "AUTHOR_ID_AUTHOR_SEQ";

CREATE trigger "BI_AUTHOR_ID_AUTHOR"
   before insert on "Author"
   for each row
begin
   select "AUTHOR_ID_AUTHOR_SEQ".nextval into :NEW."id_author" from dual;
end;
/

CREATE TABLE "Country" (
        "id_country" INT PRIMARY KEY,
        "name_country" VARCHAR2(255) UNIQUE NOT NULL);

CREATE sequence "COUNTRY_ID_COUNTRY_SEQ";

CREATE trigger "BI_COUNTRY_ID_COUNTRY"
   before insert on "Country"
```

```
    for each row
begin
    select "COUNTRY_ID_COUNTRY_SEQ".nextval into :NEW."id_country"
from dual;
end;
/

CREATE TABLE "Genre" (
        "id_genre" INT PRIMARY KEY,
        "name_genre" VARCHAR2(255) UNIQUE NOT NULL);

CREATE sequence "GENRE_ID_GENRE_SEQ";

CREATE trigger "BI_GENRE_ID_GENRE"
    before insert on "Genre"
    for each row
begin
    select "GENRE_ID_GENRE_SEQ".nextval into :NEW."id_genre" from dual;
end;
/
```

```
CREATE TABLE "Age_category" (
        "id_age_category" INT PRIMARY KEY,
        "name_age_category" VARCHAR2(255) UNIQUE NOT NULL);

CREATE sequence "AGE_CATEGORY_ID_AGE_CATEGORY_SEQ";

CREATE trigger "BI_AGE_CATEGORY_ID_AGE_CATEGORY"
   before insert on "Age_category"
   for each row
begin
   select "AGE_CATEGORY_ID_AGE_CATEGORY_SEQ".nextval into
:NEW."id_age_category" from dual;
end;
/

CREATE TABLE "Ticket" (
        "id_ticket" INT PRIMARY KEY,
        "id_performance" INT NOT NULL,
        "seat_number_ticket" INT NOT NULL CHECK("seat_number_ticket" >=
0),
```

```
        "cost_ticket" NUMERIC NOT NULL CHECK("cost_ticket" >= 0),
    "is_sold" INT NOT NULL,
    "date_sale" DATE);

CREATE sequence "TICKET_ID_TICKET_SEQ";
/

CREATE TABLE "Repertoire" (
        "id_performance" INT PRIMARY KEY,
        "id_show" INT NOT NULL,
        "performance_date_repertoire" DATE NOT NULL);

CREATE sequence "REPERTOIRE_ID_PERFORMANCE_SEQ";
/

CREATE TABLE "Direction" (
        "id_actor" INT NOT NULL,
        "id_role" INT NOT NULL,
        "is_understudy_direction" INT NOT NULL,
        constraint DIRECTION_PK PRIMARY KEY ("id_actor","id_role"));
```

```
/

CREATE TABLE "Tour" (
        "id_employee" INT NOT NULL,
        "id_show" INT NOT NULL,
        "from_date_tour" DATE NOT NULL,
        "to_date_tour" DATE NOT NULL,
        "is_visiting_tour" INT NOT NULL);
/

CREATE TABLE "Job_types" (
        "id_job_type" INT PRIMARY KEY,
    "id_parent_job_type" INT,
        "name_job_type" VARCHAR2(255) UNIQUE NOT NULL);

CREATE sequence "JOB_TYPES_ID_JOB_TYPE_SEQ";

CREATE trigger "BI_JOB_TYPES_ID_JOB_TYPE"
    before insert on "Job_types"
    for each row
```

```
begin
   select "JOB_TYPES_ID_JOB_TYPE_SEQ".nextval into :NEW."id_job_type"
from dual;
end;
/

CREATE TABLE "Role-Characteristic" (
        "id_characteristic" INT NOT NULL,
        "id_role" INT NOT NULL,
        "value_role_characteristic" DECIMAL,
        constraint ROLE_CHARACTERISTIC_PK PRIMARY KEY
("id_characteristic","id_role"));

CREATE TABLE "Role" (
        "id_role" INT PRIMARY KEY,
        "id_show" INT NOT NULL,
        "name_role" VARCHAR2(255) NOT NULL,
        "is_main_role" INT NOT NULL);

CREATE sequence "ROLE_ID_ROLE_SEQ";
```

```
CREATE trigger "BI_ROLE_ID_ROLE"
   before insert on "Role"
   for each row
begin
   select "ROLE_ID_ROLE_SEQ".nextval into :NEW."id_role" from dual;
end;
/


CREATE TABLE "Subscription" (
        "id_subscription" INT PRIMARY KEY,
        "id_genre" INT,
        "id_author" INT);


CREATE sequence "SUBSCRIPTION_ID_SUBSCRIPTION_SEQ";
/


CREATE TABLE "Ticket-Subscription" (
        "id_ticket" INT NOT NULL,
        "id_subscription" INT NOT NULL,
```

```
        constraint TICKET_SUBSCRIPTION_PK PRIMARY KEY
("id_ticket","id_subscription"));
/

CREATE TABLE "Musical_instruments" (
    "id_instrument" INT PRIMARY KEY,
    "name_instrument" VARCHAR2(255) UNIQUE NOT NULL);

CREATE sequence "MUSICAL_INSTRUMENTS_ID_INSTRUMENT_SEQ";

CREATE trigger "BI_MUSICAL_INSTRUMENTS_ID_INSTRUMENT"
    before insert on "Musical_instruments"
    for each row
begin
    select "MUSICAL_INSTRUMENTS_ID_INSTRUMENT_SEQ".nextval into
:NEW."id_instrument" from dual;
end;
/

CREATE TABLE "Musician-Instrument" (
    "id_musician" INT NOT NULL,
```

```
    "id_instrument" INT NOT NULL,
    constraint MUSICIAN_INSTRUMENT_PK PRIMARY KEY
("id_musician","id_instrument"))
/

ALTER TABLE "Employee" ADD CONSTRAINT "Employee_fk0" FOREIGN
KEY ("id_gender") REFERENCES "Gender"("id_gender");
ALTER TABLE "Employee" ADD CONSTRAINT "Employee_fk1" FOREIGN
KEY ("id_education") REFERENCES "Education"("id_education");
ALTER TABLE "Employee" ADD CONSTRAINT "Employee_fk2" FOREIGN
KEY ("id_job_type") REFERENCES "Job_types"("id_job_type");

ALTER TABLE "Musician-Show" ADD CONSTRAINT "Musician-Show_fk0"
FOREIGN KEY ("id_musician") REFERENCES "Employee"("id_employee");
ALTER TABLE "Musician-Show" ADD CONSTRAINT "Musician-Show_fk1"
FOREIGN KEY ("id_show") REFERENCES "Show"("id_show")
    ON DELETE CASCADE;

ALTER TABLE "Job_types" ADD CONSTRAINT "Job_types_fk0" FOREIGN
KEY ("id_parent_job_type") REFERENCES "Job_types"("id_job_type")
    ON DELETE CASCADE;
```

```
ALTER TABLE "Actor-Characteristic" ADD CONSTRAINT "Actor-Characteristic_fk0" FOREIGN KEY ("id_actor")
    REFERENCES "Employee"("id_employee") ON DELETE CASCADE;
ALTER TABLE "Actor-Characteristic" ADD CONSTRAINT "Actor-Characteristic_fk1" FOREIGN KEY ("id_characteristic")
    REFERENCES "Characteristic"("id_characteristic");

ALTER TABLE "Actor-Rank" ADD CONSTRAINT "Actor-Rank_fk0" FOREIGN KEY ("id_actor") REFERENCES "Employee"("id_employee")
    ON DELETE CASCADE;
ALTER TABLE "Actor-Rank" ADD CONSTRAINT "Actor-Rank_fk1" FOREIGN KEY ("id_rank") REFERENCES "Rank"("id_rank");
ALTER TABLE "Actor-Rank" ADD CONSTRAINT "Actor-Rank_fk2" FOREIGN KEY ("id_competition") REFERENCES "Competition"("id_competition");

ALTER TABLE "Show" ADD CONSTRAINT "Show_fk0" FOREIGN KEY ("id_director") REFERENCES "Employee"("id_employee");
ALTER TABLE "Show" ADD CONSTRAINT "Show_fk1" FOREIGN KEY ("id_production_designer") REFERENCES "Employee"("id_employee");
ALTER TABLE "Show" ADD CONSTRAINT "Show_fk2" FOREIGN KEY ("id_conductor") REFERENCES "Employee"("id_employee");
```

```
ALTER TABLE "Show" ADD CONSTRAINT "Show_fk3" FOREIGN KEY ("id_author") REFERENCES "Author"("id_author");

ALTER TABLE "Show" ADD CONSTRAINT "Show_fk4" FOREIGN KEY ("id_genre") REFERENCES "Genre"("id_genre");

ALTER TABLE "Show" ADD CONSTRAINT "Show_fk5" FOREIGN KEY ("id_age_category") REFERENCES "Age_category"("id_age_category");


ALTER TABLE "Author" ADD CONSTRAINT "Author_fk0" FOREIGN KEY ("id_country") REFERENCES "Country"("id_country");


ALTER TABLE "Ticket" ADD CONSTRAINT "Ticket_fk0" FOREIGN KEY ("id_performance") REFERENCES "Repertoire"("id_performance");


ALTER TABLE "Repertoire" ADD CONSTRAINT "Repertoire_fk0" FOREIGN KEY ("id_show") REFERENCES "Show"("id_show");


ALTER TABLE "Direction" ADD CONSTRAINT "Direction_fk0" FOREIGN KEY ("id_actor") REFERENCES "Employee"("id_employee");

ALTER TABLE "Direction" ADD CONSTRAINT "Direction_fk1" FOREIGN KEY ("id_role") REFERENCES "Role"("id_role");
```

```
ALTER TABLE "Tour" ADD CONSTRAINT "Tour_fk0" FOREIGN KEY ("id_employee") REFERENCES "Employee"("id_employee");

ALTER TABLE "Tour" ADD CONSTRAINT "Tour_fk1" FOREIGN KEY ("id_show") REFERENCES "Show"("id_show");


ALTER TABLE "Role-Characteristic" ADD CONSTRAINT "Role-Characteristic_fk0" FOREIGN KEY ("id_characteristic")
    REFERENCES "Characteristic"("id_characteristic");

ALTER TABLE "Role-Characteristic" ADD CONSTRAINT "Role-Characteristic_fk1" FOREIGN KEY ("id_role")
    REFERENCES "Role"("id_role") ON DELETE CASCADE;


ALTER TABLE "Role" ADD CONSTRAINT "Role_fk0" FOREIGN KEY ("id_show") REFERENCES "Show"("id_show") ON DELETE CASCADE;


ALTER TABLE "Subscription" ADD CONSTRAINT "Subscription_fk0" FOREIGN KEY ("id_genre") REFERENCES "Genre"("id_genre")
    ON DELETE CASCADE;

ALTER TABLE "Subscription" ADD CONSTRAINT "Subscription_fk1" FOREIGN KEY ("id_author") REFERENCES "Author"("id_author")
    ON DELETE CASCADE;
```

```
ALTER TABLE "Ticket-Subscription" ADD CONSTRAINT "Ticket-Subscription_fk0" FOREIGN KEY ("id_ticket")
    REFERENCES "Ticket"("id_ticket") ON DELETE CASCADE;
ALTER TABLE "Ticket-Subscription" ADD CONSTRAINT "Ticket-Subscription_fk1" FOREIGN KEY ("id_subscription")
    REFERENCES "Subscription"("id_subscription") ON DELETE CASCADE;


ALTER TABLE "Musician-Instrument" ADD CONSTRAINT "Musician-Instrument_fk0" FOREIGN KEY ("id_musician")
    REFERENCES "Employee"("id_employee") ON DELETE CASCADE;
ALTER TABLE "Musician-Instrument" ADD CONSTRAINT "Musician-Instrument_fk1" FOREIGN KEY ("id_instrument")
    REFERENCES "Musical_instruments"("id_instrument");
/


COMMIT;
```

ПРИЛОЖЕНИЕ 2. СКРИПТ УДАЛЕНИЯ СХЕМЫ БД.

```
ALTER TABLE "Employee" DROP CONSTRAINT "Employee_fk0";
ALTER TABLE "Employee" DROP CONSTRAINT "Employee_fk1";
ALTER TABLE "Employee" DROP CONSTRAINT "Employee_fk2";

ALTER TABLE "Musician-Show" DROP CONSTRAINT "Musician-Show_fk0";
```

```
ALTER TABLE "Musician-Show" DROP CONSTRAINT "Musician-Show_fk1";

ALTER TABLE "Musician-Instrument" DROP CONSTRAINT "Musician-Instrument_fk0";

ALTER TABLE "Musician-Instrument" DROP CONSTRAINT "Musician-Instrument_fk1";

ALTER TABLE "Actor-Characteristic" DROP CONSTRAINT "Actor-Characteristic_fk0";

ALTER TABLE "Actor-Characteristic" DROP CONSTRAINT "Actor-Characteristic_fk1";

ALTER TABLE "Actor-Rank" DROP CONSTRAINT "Actor-Rank_fk0";

ALTER TABLE "Actor-Rank" DROP CONSTRAINT "Actor-Rank_fk1";

ALTER TABLE "Actor-Rank" DROP CONSTRAINT "Actor-Rank_fk2";

ALTER TABLE "Show" DROP CONSTRAINT "Show_fk0";

ALTER TABLE "Show" DROP CONSTRAINT "Show_fk1";

ALTER TABLE "Show" DROP CONSTRAINT "Show_fk2";

ALTER TABLE "Show" DROP CONSTRAINT "Show_fk3";

ALTER TABLE "Show" DROP CONSTRAINT "Show_fk4";

ALTER TABLE "Show" DROP CONSTRAINT "Show_fk5";

ALTER TABLE "Author" DROP CONSTRAINT "Author_fk0";

ALTER TABLE "Ticket" DROP CONSTRAINT "Ticket_fk0";
```

```
ALTER TABLE "Repertoire" DROP CONSTRAINT "Repertoire_fk0";

ALTER TABLE "Direction" DROP CONSTRAINT "Direction_fk0";
ALTER TABLE "Direction" DROP CONSTRAINT "Direction_fk1";

ALTER TABLE "Tour" DROP CONSTRAINT "Tour_fk0";
ALTER TABLE "Tour" DROP CONSTRAINT "Tour_fk1";

ALTER TABLE "Role-Characteristic" DROP CONSTRAINT "Role-Characteristic_fk0";
ALTER TABLE "Role-Characteristic" DROP CONSTRAINT "Role-Characteristic_fk1";

ALTER TABLE "Role" DROP CONSTRAINT "Role_fk0";

ALTER TABLE "Subscription" DROP CONSTRAINT "Subscription_fk0";
ALTER TABLE "Subscription" DROP CONSTRAINT "Subscription_fk1";

ALTER TABLE "Ticket-Subscription" DROP CONSTRAINT "Ticket-Subscription_fk0";
ALTER TABLE "Ticket-Subscription" DROP CONSTRAINT "Ticket-Subscription_fk1";

DROP TABLE "Rank";
DROP TABLE "Musical_instruments";
DROP TABLE "Musician-Instrument";
DROP TABLE "Employee";
```

```
DROP TABLE "Musician-Show";
DROP TABLE "Gender";
DROP TABLE "Actor-Characteristic";
DROP TABLE "Competition";
DROP TABLE "Actor-Rank";
DROP TABLE "Characteristic";
DROP TABLE "Education";
DROP TABLE "Show";
DROP TABLE "Author";
DROP TABLE "Country";
DROP TABLE "Genre";
DROP TABLE "Age_category";
DROP TABLE "Ticket";
DROP TABLE "Repertoire";
DROP TABLE "Direction";
DROP TABLE "Tour";
DROP TABLE "Job_types";
DROP TABLE "Role-Characteristic";
DROP TABLE "Role";
DROP TABLE "Ticket-Subscription";
DROP TABLE "Subscription";

DROP sequence "RANK_ID_RANK_SEQ";
DROP sequence "EMPLOYEE_ID_EMPLOYEE_SEQ";
DROP sequence "GENDER_ID_GENDER_SEQ";
```

```
DROP sequence "COMPETITION_ID_COMPETITON_SEQ";
DROP sequence "CHARACTERISTIC_ID_CHARACTERISTIC_SEQ";
DROP sequence "EDUCATION_ID_EDUCATION_SEQ";
DROP sequence "SHOW_ID_SHOW_SEQ";
DROP sequence "AUTHOR_ID_AUTHOR_SEQ";
DROP sequence "COUNTRY_ID_COUNTRY_SEQ";
DROP sequence "GENRE_ID_GENRE_SEQ";
DROP sequence "AGE_CATEGORY_ID_AGE_CATEGORY_SEQ";
DROP sequence "TICKET_ID_TICKET_SEQ";
DROP sequence "REPERTOIRE_ID_PERFORMANCE_SEQ";
DROP sequence "JOB_TYPES_ID_JOB_TYPE_SEQ";
DROP sequence "ROLE_ID_ROLE_SEQ";
DROP sequence "SUBSCRIPTION_ID_SUBSCRIPTION_SEQ";
DROP sequence "MUSICAL_INSTRUMENTS_ID_INSTRUMENT_SEQ";

COMMIT;
```

ПРИЛОЖЕНИЕ 3. СКРИПТЫ СОЗДАНИЯ ТРИГГЕРОВ ДЛЯ ОГРАНИЧЕНИЯ ЦЕЛОСТНОСТИ БД.

```
CREATE OR REPLACE trigger "ACTOR-RANK-INSERT-UPDATE"
    before insert or update on "Actor-Rank"
    for each row
declare
    birthday DATE;
begin
    select "birthday_employee" into birthday
    from "Employee"
```

```
    where "id_employee" = :NEW."id_actor";


    if :NEW."obtaining_date_actor_rank" < birthday then
        raise_application_error(-20000, 'Получение звания не может произойти раньше рождения!');
    end if;
end;
/


CREATE OR REPLACE trigger "DIRECTION-INSERT"
    before insert on "Direction"
    for each row
declare
    actors_count INT;
    role_row "Role"%ROWTYPE;
    actor_roles INT;
    main_actors_count INT;
    today_date DATE;
    performance_cnt INT;
begin
    select CURRENT_DATE into today_date from dual;


    select * into role_row
    from "Role"
    where "id_role" = :NEW."id_role";
```

```
    select count(*) into performance_cnt
    from "Repertoire"
    where "id_show" = role_row."id_show"
        and "performance_date_repertoire" > today_date;

    if performance_cnt != 0 then
        raise_application_error(-20001, 'Нельзя утверждать новые роли, пока не пройдут показы спектакля!');
    end if;

    select count(*) into actors_count
    from "Direction"
    where "id_role" = :NEW."id_role";

    if role_row."is_main_role" = 0 and actors_count = 1 then
        raise_application_error(-20002, 'Попытка назначения второго актера на не главную роль!');
    elsif role_row."is_main_role" = 1 and actors_count = 2 then
        raise_application_error(-20003, 'Попытка назначения третьего актера на главную роль!');
    elsif role_row."is_main_role" = 1 and actors_count = 1 then
        select count(*) into main_actors_count
        from "Direction"
        where "id_role" = :NEW."id_role" and "is_understudy_direction" = :NEW."is_understudy_direction";
        if main_actors_count = 1 then
```

```
            raise_application_error(-20004, 'Попытка назначения второго основного актера или дублера на главную роль!');
        end if;
    end if;

    select count(*) into actor_roles
    from ("Direction" inner join "Role" using("id_role"))
    where "id_actor" = :NEW."id_actor" and "id_show" = role_row."id_show";

    if actor_roles != 0 then
        raise_application_error(-20005, 'Попытка назначения актера на две роли в одном и том же спектакле!');
    end if;
end;
/

CREATE OR REPLACE trigger "DIRECTION-DELETE"
    before delete on "Direction"
    for each row
declare
    id_show INT;
    performance_cnt INT;
    today_date DATE;
begin
    select CURRENT_DATE into today_date from dual;
```

```
    select "id_show" into id_show
    from "Role"
    where "id_role" = :OLD."id_role";

    select count(*) into performance_cnt
    from "Repertoire"
    where "id_show" = id_show
        and "performance_date_repertoire" > today_date;

    if performance_cnt != 0 then
        raise_application_error(-20006, 'Нельзя удалять утвержденные роли, пока не пройдут показы спектакля!');
    end if;
end;
/

CREATE OR REPLACE trigger "TOUR-INSERT"
    before insert on "Tour"
    for each row
declare
    today_date DATE;
    employee_job_type VARCHAR2(255);
    counter_director INT;
    counter_musician INT;
    counter_actor INT;
    cursor tour_cur
```

```
    is
        select *
        from "Tour"
        where "id_employee" = :NEW."id_employee";
begin
    select "name_job_type" into employee_job_type
    from ("Employee" inner join "Job_types" using("id_job_type"))
    where "id_employee" = :NEW."id_employee";

    if employee_job_type != 'актер'
        and employee_job_type != 'музыкант'
        and employee_job_type != 'режиссер-постановщик'
        and employee_job_type != 'художник-постановщик'
        and employee_job_type != 'дирижер-постановщик' then
        raise_application_error(-20007, 'Человек с данной профессией не может уезжать на гастроли!');
    end if;

    select CURRENT_DATE into today_date from dual;

    if :NEW."to_date_tour" < :NEW."from_date_tour" or :NEW."from_date_tour" < today_date then
        raise_application_error(-20008, 'Дата начала гастролей позже даты конца или раньше сегодняшнего дня!');
    end if;
```

```
    select count(*) into counter_director
    from "Show"
    where ("id_director" = :NEW."id_employee"
          or "id_production_designer" = :NEW."id_employee"
          or "id_conductor" = :NEW."id_employee")
          and "id_show" = :NEW."id_show";

    select count(*) into counter_musician
    from "Musician-Show"
    where "id_musician" = :NEW."id_employee"
          and "id_show" = :NEW."id_show";

    select count(*) into counter_actor
    from ("Direction" inner join "Role" using("id_role"))
    where "id_actor" = :NEW."id_employee"
          and "id_show" = :NEW."id_show";

    if counter_actor + counter_director + counter_musician = 0 then
       raise_application_error(-20009, 'Этот человек не участвует в этом
спектакле!');
    end if;

    for tour_rec in tour_cur
    loop
       if tour_rec."from_date_tour" <= :NEW."from_date_tour" and
tour_rec."to_date_tour" >= :NEW."from_date_tour"
```

```
            or tour_rec."from_date_tour" <= :NEW."to_date_tour" and
tour_rec."to_date_tour" >= :NEW."to_date_tour" then
                raise_application_error(-20010, 'У этого сотрудника есть
пересекающиеся гастроли!');
        end if;
    end loop;
end;
/


CREATE OR REPLACE trigger "SHOW-INSERT-UPDATE"
    before insert or update on "Show"
    for each row
declare
    author_century INT;
    today_date DATE;
    performance_cnt INT;
    cursor repertoire_cur
    is
        select *
        from "Repertoire"
        where "id_show" = :NEW."id_show";
begin
    select CURRENT_DATE into today_date from dual;

    if updating then
        select count(*) into performance_cnt
```

```
        from "Repertoire"
        where "id_show" = :NEW."id_show"
            and "performance_date_repertoire" > today_date;

        if performance_cnt != 0 then
            raise_application_error(-20011, 'Нельзя модифицировать информацию, пока не пройдут показы спектакля!');
        end if;
    end if;

    select "life_century_author" into author_century
    from "Author"
    where "id_author" = :NEW."id_author";

    if :NEW."century_show" < author_century then
        raise_application_error(-20012, 'Век спектакля раньше века жизни автора!');
    end if;

    if inserting then
        if :NEW."premier_date_show" < today_date then
            raise_application_error(-20013, 'Премьера спектакля не может быть раньше сегодняшней даты!');
        end if;
    else
        if :NEW."premier_date_show" < :OLD."premier_date_show" then
```

```
            raise_application_error(-20014, 'Премьера спектакля не может быть раньше предыдущей даты!');
        end if;
    end if;

    for repertoire_rec in repertoire_cur
    loop
        if repertoire_rec."performance_date_repertoire" < :NEW."premier_date_show" then
            raise_application_error(-20015, 'Премьера спектакля не может быть позже даты выступления!');
        end if;
    end loop;

    if inserting then
        select "SHOW_ID_SHOW_SEQ".nextval into :NEW."id_show" from dual;
    end if;
end;
/

CREATE OR REPLACE trigger "SHOW-DELETE"
    before delete on "Show"
    for each row
declare
    cnt INT;
begin
```

```
    select count(*) into cnt
    from "Repertoire"
    where "id_show" = :OLD."id_show";

    if cnt != 0 then
        raise_application_error(-20016, 'Этот спектакль уже в репертуаре!');
    end if;
end;
/

CREATE OR REPLACE trigger "MUSICIAN-SHOW-DELETE"
    before delete on "Musician-Show"
    for each row
declare
    cnt INT;
    today_date DATE;
begin
    select CURRENT_DATE into today_date from dual;

    select count(*) into cnt
    from "Repertoire"
    where "id_show" = :OLD."id_show"
        and "performance_date_repertoire" > today_date;

    if cnt != 0 then
```

```
        raise_application_error(-20017, 'Этот спектакль уже в репертуаре!');
    end if;
end;
/

CREATE OR REPLACE trigger "MUSICIAN-SHOW-INSERT"
    before insert on "Musician-Show"
    for each row
declare
    cnt INT;
    today_date DATE;
begin
    select CURRENT_DATE into today_date from dual;

    select count(*) into cnt
    from "Repertoire"
    where "id_show" = :NEW."id_show"
        and "performance_date_repertoire" > today_date;

    if cnt != 0 then
        raise_application_error(-20018, 'Этот спектакль уже в репертуаре!');
    end if;
end;
/
```

```
CREATE OR REPLACE trigger "REPERTOIRE-INSERT"
    before insert on "Repertoire"
    for each row
declare
    premier_date DATE;
    actors_count INT;
    cursor show_roles is
        select "id_role", "is_main_role"
        from "Role"
        where "id_show" = :NEW."id_show";
    performance_amount INT;


begin
    select "premier_date_show" into premier_date
    from "Show"
    where "id_show" = :NEW."id_show";


    if :NEW."performance_date_repertoire" < premier_date then
        raise_application_error(-20019, 'Дата показа не может быть раньше даты премьеры!');
    end if;


    for role_record in show_roles
    loop
        select count(*) into actors_count
        from "Direction"
```

```
        where "id_role" = role_record."id_role";


        if role_record."is_main_role" = 0 and actors_count < 1 or
role_record."is_main_role" = 1 and actors_count < 2 then
            raise_application_error(-20020, 'Актерский состав для данного
спектакля сформирован не полностью!');
        end if;
    end loop;


    select count(*) into performance_amount
    from "Repertoire"
    where "performance_date_repertoire" = :NEW."performance_date_repertoire";


    if performance_amount != 0 then
        raise_application_error(-20021, 'На это время уже назначен спектакль!');
    end if;


    select "REPERTOIRE_ID_PERFORMANCE_SEQ".nextval into
:NEW."id_performance" from dual;
end;
/


CREATE OR REPLACE trigger "REPERTOIRE-DELETE"
    before delete on "Repertoire"
    for each row
declare
```

```
    tickets_amount INT;
begin
    select count(*) into tickets_amount
    from "Ticket"
    where "id_performance" = :OLD."id_performance";

    if tickets_amount != 0 then
        raise_application_error(-20022, 'На это выступление уже выпущены билеты!');
    end if;
end;
/

CREATE OR REPLACE trigger "TICKET-INSERT"
    before insert on "Ticket"
    for each row
declare
    tickets_count INT;
begin
    select count(*) into tickets_count
    from "Ticket"
    where "id_performance" = :NEW."id_performance" and "seat_number_ticket" = :NEW."seat_number_ticket";
    if tickets_count = 1 then
        raise_application_error(-20023, 'Билет на это место для этого показа уже существует!');
```

```
    end if;

    select "TICKET_ID_TICKET_SEQ".nextval into :NEW."id_ticket" from dual;
end;
/

CREATE OR REPLACE trigger "TICKET-SUBSCRIPTION-INSERT"
    before insert on "Ticket-Subscription"
    for each row
declare
    tickets_count INT;
    id_show INT;
    id_show_author INT;
    id_show_genre INT;
    id_subscription_author INT;
    id_subscription_genre INT;
begin
    select count(*) into tickets_count
    from "Ticket-Subscription"
    where "id_ticket" = :NEW."id_ticket";

    if tickets_count = 1 then
        raise_application_error(-20024, 'Этот билет уже добавлен в абонемент!');
    end if;
```

```
    select "id_show" into id_show
    from ("Ticket" inner join "Repertoire" using("id_performance"))
    where "id_ticket" = :NEW."id_ticket";

    select "id_author", "id_genre" into id_show_author, id_show_genre
    from "Show"
    where "id_show" = id_show;

    select "id_author", "id_genre" into id_subscription_author,
id_subscription_genre
    from "Subscription"
    where "id_subscription" = :NEW."id_subscription";

    if id_subscription_author is null and id_subscription_genre != id_show_genre
    or id_subscription_genre is null and id_subscription_author != id_show_author
then
        raise_application_error(-20025, 'Этот билет имеет неподходящую под
абонемент тематику!');
    end if;
end;
/

CREATE OR REPLACE trigger "SUBSCRIPTION-INSERT-UPDATE"
    before insert or update on "Subscription"
    for each row
begin
```

```
    if :NEW."id_genre" is not null and :NEW."id_author" is not null
        or :NEW."id_genre" is null and :NEW."id_author" is null then
        raise_application_error(-20026, 'Абонемент может иметь либо конкретного автора, либо конкретный жанр!');
    end if;

    if inserting then
        select "SUBSCRIPTION_ID_SUBSCRIPTION_SEQ".nextval into :NEW."id_subscription" from dual;
    end if;
end;
/

CREATE OR REPLACE trigger "EMPLOYEE-INSERT-UPDATE"
    before insert or update on "Employee"
    for each row
declare
    today_date DATE;
begin
    if updating or inserting then
        if :NEW."hire_date_employee" < :NEW."birthday_employee" then
            raise_application_error(-20027, 'Дата рождения сотрудника позже даты найма!');
        end if;

        select CURRENT_DATE into today_date from dual;
```

```
        if :NEW."birthday_employee" > today_date then
            raise_application_error(-20028, 'Дата рождения сотрудника позже сегоднящнего дня!');
        end if;
    end if;

    if inserting then
        select "EMPLOYEE_ID_EMPLOYEE_SEQ".nextval into :NEW."id_employee" from dual;
    end if;
end;
/

CREATE OR REPLACE trigger "EMPLOYEE-DELETE"
    before delete on "Employee"
    for each row
declare
    counter_director INT;
    counter_musician INT;
    counter_actor INT;
begin
    select count(*) into counter_director
    from "Show"
    where ("id_director" = :OLD."id_employee"
        or "id_production_designer" = :OLD."id_employee"
```

```
        or "id_conductor" = :OLD."id_employee");

    select count(*) into counter_musician
    from "Musician-Show"
    where "id_musician" = :OLD."id_employee";

    select count(*) into counter_actor
    from "Direction"
    where "id_actor" = :OLD."id_employee";

    if counter_actor + counter_director + counter_musician != 0 then
        raise_application_error(-20029, 'Этот человек занят в спектаклях!');
    end if;
end;
/

CREATE OR REPLACE trigger "CHARACTERISTIC-DELETE"
    before delete on "Characteristic"
    for each row
declare
    cnt INT;
begin
    select count(*) into cnt
    from "Actor-Characteristic"
    where "id_characteristic" = :OLD."id_characteristic";
```

```
    if cnt != 0 then
        raise_application_error(-20030, 'Данная запись используется!');
    end if;

    select count(*) into cnt
    from "Role-Characteristic"
    where "id_characteristic" = :OLD."id_characteristic";

    if cnt != 0 then
        raise_application_error(-20031, 'Данная запись используется!');
    end if;
end;
/

CREATE OR REPLACE trigger "GENDER-DELETE"
    before delete on "Gender"
    for each row
declare
    cnt INT;
begin
    select count(*) into cnt
    from "Employee"
    where "id_gender" = :OLD."id_gender";
```

```
    if cnt != 0 then
        raise_application_error(-20032, 'Данная запись используется!');
    end if;
end;
/

CREATE OR REPLACE trigger "MUSICAL_INSTRUMENTS-DELETE"
    before delete on "Musical_instruments"
    for each row
declare
    cnt INT;
begin
    select count(*) into cnt
    from "Musician-Instrument"
    where "id_instrument" = :OLD."id_instrument";

    if cnt != 0 then
        raise_application_error(-20033, 'Данная запись используется!');
    end if;
end;
/

CREATE OR REPLACE trigger "EDUCATION-DELETE"
    before delete on "Education"
    for each row
```

```
declare
    cnt INT;
begin
    select count(*) into cnt
    from "Employee"
    where "id_education" = :OLD."id_education";

    if cnt != 0 then
        raise_application_error(-20034, 'Данная запись используется!');
    end if;
end;
/

CREATE OR REPLACE trigger "JOB_TYPES-DELETE"
    before delete on "Job_types"
    for each row
declare
    cnt INT;
begin
    select count(*) into cnt
    from "Employee"
    where "id_job_type" = :OLD."id_job_type";

    if cnt != 0 then
        raise_application_error(-20035, 'Данная запись используется!');
```

```
    end if;
end;
/

CREATE OR REPLACE trigger "AGE_CATEGORY-DELETE"
    before delete on "Age_category"
    for each row
declare
    cnt INT;
begin
    select count(*) into cnt
    from "Show"
    where "id_age_category" = :OLD."id_age_category";

    if cnt != 0 then
        raise_application_error(-20036, 'Данная запись используется!');
    end if;
end;
/

CREATE OR REPLACE trigger "GENRE-DELETE"
    before delete on "Genre"
    for each row
declare
    cnt INT;
```

```
begin
    select count(*) into cnt
    from "Show"
    where "id_genre" = :OLD."id_genre";

    if cnt != 0 then
        raise_application_error(-20037, 'Данная запись используется!');
    end if;
end;
/

CREATE OR REPLACE trigger "COUNTRY-DELETE"
    before delete on "Country"
    for each row
declare
    cnt INT;
begin
    select count(*) into cnt
    from "Author"
    where "id_country" = :OLD."id_country";

    if cnt != 0 then
        raise_application_error(-20038, 'Данная запись используется!');
    end if;
end;
```

```
/

CREATE OR REPLACE trigger "RANK-DELETE"
    before delete on "Rank"
    for each row
declare
    cnt INT;
begin
    select count(*) into cnt
    from "Actor-Rank"
    where "id_rank" = :OLD."id_rank";

    if cnt != 0 then
        raise_application_error(-20039, 'Данная запись используется!');
    end if;
end;
/

CREATE OR REPLACE trigger "COMPETITION-DELETE"
    before delete on "Competition"
    for each row
declare
    cnt INT;
begin
    select count(*) into cnt
```

```
    from "Actor-Rank"
    where "id_competition" = :OLD."id_competition";

    if cnt != 0 then
        raise_application_error(-20040, 'Данная запись используется!');
    end if;
end;
/

CREATE OR REPLACE trigger "ROLE-DELETE"
    before delete on "Role"
    for each row
declare
    cnt INT;
begin
    select count(*) into cnt
    from "Direction"
    where "id_role" = :OLD."id_role";

    if cnt != 0 then
        raise_application_error(-20041, 'Данная запись используется!');
    end if;
end;
/
```

```
CREATE OR REPLACE trigger "ROLE-CHARACTERISTIC-INSERT-
UPDATE-DELETE"
    before insert or update or delete on "Role-Characteristic"
    for each row
declare
    cnt INT;
    id_role INT;
begin
    if inserting or updating then
        id_role := :NEW."id_role";
    else
        id_role := :OLD."id_role";
    end if;
    select count(*) into cnt
    from "Direction"
    where "id_role" = id_role;

    if cnt != 0 then
        raise_application_error(-20042, 'Данная запись используется!');
    end if;
end;
/

CREATE OR REPLACE trigger "AUTHOR-DELETE"
    before delete on "Author"
    for each row
```

```
declare
    cnt INT;
begin
    select count(*) into cnt
    from "Show"
    where "id_author" = :OLD."id_author";

    if cnt != 0 then
        raise_application_error(-20043, 'Данная запись используется!');
    end if;
end;
/

CREATE OR REPLACE trigger "AUTHOR-UPDATE"
    before update on "Author"
    for each row
declare
    cursor show_cur
    is
        select "century_show"
        from "Show"
        where "id_author" = :NEW."id_author";
begin
    for show_rec in show_cur
    loop
```

```
        if show_rec."century_show" < :NEW."life_century_author" then
            raise_application_error(-20044, 'Век жизни автора не может быть позже времени постановки его спектаклей!');
        end if;
    end loop;
end;
/


COMMIT;
```

ПРИЛОЖЕНИЕ 4. СОЗДАНИЕ ХРАНИМЫХ ПРОЦЕДУР.

```
CREATE OR REPLACE procedure show_info(
    from_date_show IN VARCHAR2,
    to_date_show IN VARCHAR2,
    first_time_show IN INT,
    id_show IN INT,
    status IN INT,
    from_century_show IN INT,
    to_century_show IN INT,
    id_conductor IN INT,
    id_production_designer IN INT,
    id_director IN INT,
    id_genre IN INT,
    id_age_category IN INT,
    id_author IN INT,
    id_country IN INT,
```

```
    show_cur OUT SYS_REFCURSOR)
is
begin
    open show_cur for
        select "id_show", "name_show" as "название",
            (select "name_employee" || ' ' || "surname_employee" || ' ' ||
"middle_name_employee"
            from "Employee" where "id_employee" = "id_director") as "режиссер-
постановщик",
        (select "name_employee" || ' ' || "surname_employee" || ' ' ||
"middle_name_employee"
            from "Employee" where "id_employee" = "id_conductor") as "дирижер-
постановщик",
        (select "name_employee" || ' ' || "surname_employee" || ' ' ||
"middle_name_employee"
            from "Employee" where "id_employee" = "id_production_designer") as
"художник-постановщик",
        ("name_author" || ' ' || "surname_author" || ' ' || "middle_name_author") as
"автор",
        "name_genre" as "жанр",
        "century_show" as "век",
        "premier_date_show" as "премьера"
        from (("Show" inner join "Author" using("id_author")) inner join "Genre"
using("id_genre"))
        where "id_show" in (select "id_show"
                    from (("Repertoire" inner join "Show" using ("id_show"))
                        inner join "Author" using ("id_author"))
                    where
```

```
                    ("performance_date_repertoire" <=
NVL(TO_DATE(to_date_show, 'yyyy/mm/dd'),
                        "performance_date_repertoire")
                    and "performance_date_repertoire" >=
NVL(TO_DATE(from_date_show, 'yyyy/mm/dd'),
                        "performance_date_repertoire")
                    and ((status = 0) or "performance_date_repertoire" <= (select
CURRENT_DATE from dual)))


                    and "id_show" = NVL(id_show, "id_show")
                    and "century_show" <= NVL(to_century_show,
"century_show")
                        and "century_show" >= NVL(from_century_show,
"century_show")
                    and "id_conductor" = NVL(id_conductor, "id_conductor")
                    and "id_production_designer" = NVL(id_production_designer,
"id_production_designer")
                    and "id_director" = NVL(id_director, "id_director")
                    and "id_genre" = NVL(id_genre, "id_genre")
                    and "id_age_category" = NVL(id_age_category,
"id_age_category")
                    and "id_author" = NVL(id_author, "id_author")
                    and "id_country" = NVL(id_country, "id_country")
                group by "id_show"
                having first_time_show = 0 or (first_time_show != 0 and
count(*) = 1));
end;
/
```

```
CREATE OR REPLACE procedure author_info(
    name_author IN VARCHAR2,
    surname_author IN VARCHAR2,
    middle_name_author IN VARCHAR2,
    from_century_life IN INT,
    to_century_life IN INT,
    id_country IN INT,
    author_cur OUT SYS_REFCURSOR)
is
begin
    open author_cur for
        select "id_author",
        "name_author" as "имя",  "surname_author" as "фаммлия", "middle_name_author" as "отчество",
        "life_century_author" as "век жизни",
        "name_country" as "страна"
        from ("Author" inner join "Country" using("id_country"))
        where ("name_author" like '%' || name_author || '%' or name_author is null)
            and ("surname_author" like '%' || surname_author || '%' or surname_author is null)
            and ("middle_name_author" like '%' || middle_name_author || '%' or middle_name_author is null)
            and "life_century_author" >= NVL(from_century_life, "life_century_author")
            and "life_century_author" <= NVL(to_century_life, "life_century_author")
            and "id_country" = NVL(id_country, "id_country");
```

```
end;
/

CREATE OR REPLACE procedure author_insert(
    name IN "Author"."name_author"%TYPE,
    surname IN "Author"."surname_author"%TYPE,
    middle_name IN "Author"."middle_name_author"%TYPE,
    century IN "Author"."life_century_author"%TYPE,
    country IN "Author"."id_country"%TYPE)
is
begin
    INSERT INTO "Author"("id_author", "name_author", "surname_author",
"middle_name_author", "life_century_author", "id_country")
        VALUES(0, name, surname, middle_name, century, country);
    COMMIT;
end;
/

CREATE OR REPLACE procedure author_update(
    name IN "Author"."name_author"%TYPE,
    surname IN "Author"."surname_author"%TYPE,
    middle_name IN "Author"."middle_name_author"%TYPE,
    century IN "Author"."life_century_author"%TYPE,
    country IN "Author"."id_country"%TYPE,
    id_author IN "Author"."id_author"%TYPE)
is
```

```
begin
   UPDATE "Author" SET "name_author" = name, "surname_author" = surname,
"middle_name_author" = middle_name,
      "life_century_author" = century, "id_country" = country
   WHERE "id_author" = id_author;
   COMMIT;
end;
/

CREATE OR REPLACE procedure author_delete(
   id_author IN "Author"."id_author"%TYPE)
is
begin
   DELETE FROM "Author" WHERE "id_author" = id_author;
   COMMIT;
end;
/

CREATE OR REPLACE procedure author_shows(
   from_date_show IN "Repertoire"."performance_date_repertoire"%TYPE,
   to_date_show IN "Repertoire"."performance_date_repertoire"%TYPE,
   id_genre IN "Genre"."id_genre"%TYPE,
   id_author IN "Author"."id_author"%TYPE,
   show_cur OUT SYS_REFCURSOR)
is
begin
```

```
open show_cur for
    select "id_show", "name_show" as "название",
        (select "name_employee" || ' ' || "surname_employee" || ' ' ||
"middle_name_employee"
        from "Employee" where "id_employee" = "id_director") as "режиссер-
постановщик",
    (select "name_employee" || ' ' || "surname_employee" || ' ' ||
"middle_name_employee"
        from "Employee" where "id_employee" = "id_conductor") as "дирижер-
постановщик",
    (select "name_employee" || ' ' || "surname_employee" || ' ' ||
"middle_name_employee"
        from "Employee" where "id_employee" = "id_production_designer") as
"художник-постановщик",
    ("name_author" || ' ' || "surname_author" || ' ' || "middle_name_author") as
"автор",
    "name_genre" as "жанр",
    "century_show" as "век",
    "premier_date_show" as "премьера"
    from (("Show" inner join "Author" using("id_author")) inner join "Genre"
using("id_genre"))
    where "id_show" in (select "id_show"
                from (("Repertoire" inner join "Show" using ("id_show"))
                    inner join "Author" using ("id_author"))
                where
                    ("performance_date_repertoire" <=
NVL(TO_DATE(to_date_show, 'yyyy/mm/dd'),
                        "performance_date_repertoire")
```

```
                        and "performance_date_repertoire" >=
NVL(TO_DATE(from_date_show, 'yyyy/mm/dd'),
                            "performance_date_repertoire")


                        and "id_genre" = NVL(id_genre, "id_genre")
                        and "id_author" = NVL(id_author, "id_author"))
                    group by "id_show");


end;
/


CREATE OR REPLACE procedure actor_role_show_info(id_show IN INT, list
OUT SYS_REFCURSOR)
is
begin
    open list for
    select ("name_employee" || ' ' || "surname_employee" || ' ' ||
"middle_name_employee") as "актер",
        "name_role" as "роль", "is_understudy_direction" as "дублер"
    from ("Direction" inner join "Employee" on "id_actor" = "id_employee")
        inner join "Role" using("id_role")
    where "id_show" = id_show;
end;
/


CREATE OR REPLACE procedure musician_show_info(id_show IN INT, list
OUT SYS_REFCURSOR)
```

```
is
begin
    open list for
    select ("name_employee" || ' ' || "surname_employee" || ' ' ||
"middle_name_employee") as "музыкант",
        "name_instrument" as "инструмент"
    from (("Musician-Show" inner join "Employee" on "id_musician" =
"id_employee")
        inner join "Musician-Instrument" using("id_musician"))
        inner join "Musical_instruments" using("id_instrument")
    where "id_show" = id_show;
end;
/

CREATE OR REPLACE procedure get_genders_list(list OUT
SYS_REFCURSOR)
is
begin
    open list for
    select "id_gender", "name_gender"
    from "Gender";
end;
/

CREATE OR REPLACE procedure get_education_list(list OUT
SYS_REFCURSOR)
is
```

```
begin
    open list for
    select "id_education", "name_education"
    from "Education";
end;
/

CREATE OR REPLACE procedure get_job_types_list(list OUT
SYS_REFCURSOR)
is
begin
    open list for
    select "id_job_type", "name_job_type"
    from "Job_types";
end;
/

CREATE OR REPLACE procedure employee_info(
    name IN "Employee"."name_employee"%TYPE,
    surname IN "Employee"."surname_employee"%TYPE,
    middle_name IN "Employee"."middle_name_employee"%TYPE,
    id_gender IN "Employee"."id_gender"%TYPE,
    birthday_from IN "Employee"."birthday_employee"%TYPE,
    birthday_to IN "Employee"."birthday_employee"%TYPE,
    age_from IN INT,
    age_to IN INT,
```

```
    experience_from IN INT,
    experience_to IN INT,
    children_amount_from IN "Employee"."children_amount_employee"%TYPE,
    children_amount_to IN "Employee"."children_amount_employee"%TYPE,
    salary_from IN "Employee"."salary_employee"%TYPE,
    salary_to IN "Employee"."salary_employee"%TYPE,
    id_education IN "Employee"."id_education"%TYPE,
    id_job_type IN "Employee"."id_job_type"%TYPE,
    employee_cur OUT SYS_REFCURSOR
)
is
begin
    open employee_cur for
        select
            "id_employee",
            "name_employee" as "имя",
            "surname_employee" as "фамилия",
            "middle_name_employee" as "отчество",
            "name_gender" as "гендер",
            "birthday_employee" as "дата рождения",
            "hire_date_employee" as "дата найма",
            "children_amount_employee" as "кол-во детей",
            "salary_employee" as "зарплата(руб.)",
            "name_education" as "образование",
            "name_job_type" as "должность"
```

```
    from ((("Employee" inner join "Education" using("id_education"))
        inner join "Gender" using("id_gender"))
        inner join "Job_types" using("id_job_type"))
    where ("name_employee" like '%' || name || '%' or name is null)
        and ("surname_employee" like '%' || surname || '%' or surname is null)
        and ("middle_name_employee" like '%' || middle_name || '%' or
middle_name is null)
        and "id_gender" = NVL(id_gender, "id_gender")
        and "birthday_employee" >= NVL(birthday_from, "birthday_employee")
        and "birthday_employee" <= NVL(birthday_to, "birthday_employee")
        and TRUNC(((SELECT SYSDATE FROM DUAL) -
"hire_date_employee")) >= NVL(experience_from,
                                    TRUNC(((SELECT SYSDATE FROM
DUAL) - "hire_date_employee")))
        and TRUNC(((SELECT SYSDATE FROM DUAL) -
"hire_date_employee")) <= NVL(experience_to,
                                    TRUNC(((SELECT SYSDATE FROM
DUAL) - "hire_date_employee")))
        and TRUNC(((SELECT SYSDATE FROM DUAL) -
"birthday_employee")) >= NVL(age_from,
                                    TRUNC(((SELECT SYSDATE FROM
DUAL) - "birthday_employee")))
        and TRUNC(((SELECT SYSDATE FROM DUAL) -
"birthday_employee")) <= NVL(age_to,
                                    TRUNC(((SELECT SYSDATE FROM
DUAL) - "birthday_employee")))
        and "children_amount_employee" >= NVL(children_amount_from,
"children_amount_employee")
```

```
        and "children_amount_employee" <= NVL(children_amount_to,
"children_amount_employee")
        and "salary_employee" >= NVL(salary_from, "salary_employee")
        and "salary_employee" <= NVL(salary_to, "salary_employee")
        and "id_education" = NVL(id_education, "id_education")
        and (id_job_type is null or is_sub_job_type(id_job_type, "id_job_type") =
1);
end;
/

CREATE OR REPLACE function is_sub_job_type(
   id_parent_job_type IN "Job_types"."id_parent_job_type"%TYPE,
   id_job_type IN "Job_types"."id_job_type"%TYPE
)
return int
is
   cursor job_cur is
      select "id_job_type"
      from "Job_types"
      where "id_parent_job_type" = id_parent_job_type;

   result int;
begin
   if id_parent_job_type = id_job_type then
      return 1;
   end if;
```

```
    for job_rec in job_cur
    loop
        if job_rec."id_job_type" = id_job_type then
            return 1;
        else
            result := is_sub_job_type(job_rec."id_job_type", id_job_type);
            if result = 1 then
                return 1;
            end if;
        end if;
    end loop;

    return 0;
end;
/

CREATE OR REPLACE procedure employee_insert(
    name IN "Employee"."name_employee"%TYPE,
    surname IN "Employee"."surname_employee"%TYPE,
    middle_name IN "Employee"."middle_name_employee"%TYPE,
    id_gender IN "Employee"."id_gender"%TYPE,
    birthday IN "Employee"."birthday_employee"%TYPE,
    hire_date IN "Employee"."hire_date_employee"%TYPE,
    children_amount IN "Employee"."children_amount_employee"%TYPE,
```

```
    salary IN "Employee"."salary_employee"%TYPE,
    id_education IN "Employee"."id_education"%TYPE,
    id_job_type IN "Employee"."id_job_type"%TYPE)
is
begin
    INSERT INTO "Employee" VALUES(0, name, surname, middle_name,
id_gender, birthday, hire_date, children_amount, salary,
        id_education, id_job_type);
    COMMIT;
end;
/

CREATE OR REPLACE procedure employee_update(
    name IN "Employee"."name_employee"%TYPE,
    surname IN "Employee"."surname_employee"%TYPE,
    middle_name IN "Employee"."middle_name_employee"%TYPE,
    id_gender IN "Employee"."id_gender"%TYPE,
    birthday IN "Employee"."birthday_employee"%TYPE,
    hire_date IN "Employee"."hire_date_employee"%TYPE,
    children_amount IN "Employee"."children_amount_employee"%TYPE,
    salary IN "Employee"."salary_employee"%TYPE,
    id_education IN "Employee"."id_education"%TYPE,
    id_job_type IN "Employee"."id_job_type"%TYPE,
    id_employee IN "Employee"."id_employee"%TYPE)
is
begin
```

```
    UPDATE "Employee" SET "name_employee" = name, "surname_employee" =
surname, "middle_name_employee" = middle_name,
        "id_gender" = id_gender, "birthday_employee" = birthday,
"hire_date_employee" = hire_date,
        "children_amount_employee" = children_amount, "salary_employee" =
salary, "id_education" = id_education,
        "id_job_type" = id_job_type
    WHERE "id_employee" = id_employee;
    COMMIT;
end;
/

CREATE OR REPLACE procedure employee_delete(
    id_employee IN "Employee"."id_employee"%TYPE)
is
begin
    DELETE FROM "Employee" WHERE "id_employee" = id_employee;
    COMMIT;
end;
/

CREATE OR REPLACE procedure get_shows_list(list OUT SYS_REFCURSOR)
is
begin
    open list for
    select "id_show", "name_show"
```

```
    from "Show";
end;
/

CREATE OR REPLACE procedure get_genres_list(list OUT SYS_REFCURSOR)
is
begin
    open list for
    select "id_genre", "name_genre"
    from "Genre";
end;
/

CREATE OR REPLACE procedure get_age_categories_list(list OUT
SYS_REFCURSOR)
is
begin
    open list for
    select "id_age_category", "name_age_category"
    from "Age_category";
end;
/

CREATE OR REPLACE procedure get_authors_list(list OUT
SYS_REFCURSOR)
is
```

```
begin
    open list for
    select "id_author", "name_author"
    from "Author";
end;
/


CREATE OR REPLACE procedure get_countries_list(list OUT
SYS_REFCURSOR)
is
begin
    open list for
    select "id_country", "name_country"
    from "Country";
end;
/


CREATE OR REPLACE procedure get_employee_list(list OUT
SYS_REFCURSOR)
is
begin
    open list for
    select "id_employee", ("name_employee" || ' ' || "surname_employee" || ' ' ||
"middle_name_employee") as name
    from ("Employee" inner join "Job_types" using("id_job_type"))
    where "name_job_type" like 'режиссер-постановщик';
```

```
end;
/

CREATE OR REPLACE procedure get_conductors_list(list OUT
SYS_REFCURSOR)
is
begin
   open list for
   select "id_employee", ("name_employee" || ' ' || "surname_employee" || ' ' ||
"middle_name_employee") as name
   from ("Employee" inner join "Job_types" using("id_job_type"))
   where "name_job_type" like 'дирижер-постановщик';
end;
/

CREATE OR REPLACE procedure get_designers_list(list OUT
SYS_REFCURSOR)
is
begin
   open list for
   select "id_employee", ("name_employee" || ' ' || "surname_employee" || ' ' ||
"middle_name_employee") as name
   from ("Employee" inner join "Job_types" using("id_job_type"))
   where "name_job_type" like 'художник-постановщик';
end;
/
```

```
create or replace procedure get_directors_list(list OUT SYS_REFCURSOR)
is
begin
    open list for
    select "id_employee", ("name_employee" || ' '  || "surname_employee" || ' ' || "middle_name_employee") as name
    from ("Employee" inner join "Job_types" using("id_job_type"))
    where "name_job_type" like 'режиссер-постановщик';
end;
/

CREATE OR REPLACE procedure get_actors_list(list OUT SYS_REFCURSOR)
is
begin
    open list for
    select "id_employee", ("name_employee" || ' ' || "surname_employee" || ' ' || "middle_name_employee") as name
    from ("Employee" inner join "Job_types" using("id_job_type"))
    where "name_job_type" like 'актер';
end;
/

CREATE OR REPLACE procedure get_rank_list(list OUT SYS_REFCURSOR)
is
begin
    open list for
```

```
    select "id_rank", "name_rank"
    from "Rank";
end;
/

CREATE OR REPLACE procedure get_competition_list(list OUT
SYS_REFCURSOR)
is
begin
    open list for
    select "id_competition", "name_competition"
    from "Competition";
end;
/

CREATE OR REPLACE procedure actor_rank_info(
    id_actor IN "Employee"."id_employee"%TYPE,
    id_gender IN "Employee"."id_gender"%TYPE,
    age_from IN INT,
    age_to IN INT,
    date_from IN "Actor-Rank"."obtaining_date_actor_rank"%TYPE,
    date_to IN "Actor-Rank"."obtaining_date_actor_rank"%TYPE,
    id_rank IN "Rank"."id_rank"%TYPE,
    id_competition IN "Competition"."id_competition"%TYPE,
    actor_rank_cur OUT SYS_REFCURSOR)
is
```

```
begin
   open actor_rank_cur for
      select "name_rank" as "звание", "name_competition" as "конкурс",
"obtaining_date_actor_rank" as "дата получения",
      ("name_employee" || ' ' || "surname_employee" || ' ' ||
"middle_name_employee") as "актер"
      from ("Actor-Rank"
         inner join "Rank" using("id_rank"))
         inner join "Competition" using("id_competition")
         inner join "Employee" on "id_actor" = "id_employee"
      where
         "id_actor" = NVL(id_actor, "id_actor")
         and "id_gender" = NVL(id_gender, "id_gender")
         and TRUNC(((SELECT SYSDATE FROM DUAL) -
"birthday_employee")) >= NVL(age_from,
                                          TRUNC(((SELECT SYSDATE FROM
DUAL) - "birthday_employee")))
         and TRUNC(((SELECT SYSDATE FROM DUAL) -
"birthday_employee")) <= NVL(age_to,
                                          TRUNC(((SELECT SYSDATE FROM
DUAL) - "birthday_employee")))
         and "obtaining_date_actor_rank" >= NVL(date_from,
"obtaining_date_actor_rank")
         and "obtaining_date_actor_rank" <= NVL(date_to,
"obtaining_date_actor_rank")
         and "id_rank" = NVL(id_rank, "id_rank")
         and "id_competition" = NVL(id_competition, "id_competition");
```

```
end;
/

CREATE OR REPLACE procedure actor_rank_list(id_actor IN "Actor-Rank"."id_actor"%TYPE, list OUT SYS_REFCURSOR)
is
begin
    open list for
        select "id_rank", "name_rank" as "звание"
        from "Actor-Rank" inner join "Rank" using("id_rank")
        where "id_rank" = id_actor;
end;
/

CREATE OR REPLACE procedure actor_rank_insert(
    id_actor IN "Actor-Rank"."id_actor"%TYPE,
    id_rank IN "Actor-Rank"."id_rank"%TYPE,
    obtaining_date IN "Actor-Rank"."obtaining_date_actor_rank"%TYPE,
    id_competition IN "Actor-Rank"."id_competition"%TYPE
)
is
begin
    INSERT INTO "Actor-Rank" VALUES(id_actor, id_rank, obtaining_date, id_competition);
end;
/
```

```
CREATE OR REPLACE procedure actor_rank_delete(
   id_actor IN "Actor-Rank"."id_actor"%TYPE,
   id_rank IN "Actor-Rank"."id_rank"%TYPE
)
is
begin
   DELETE FROM "Actor-Rank"
   WHERE "id_actor" = id_actor and "id_rank" = id_rank;
end;
/

CREATE OR REPLACE procedure actor_characteristic_insert(
   id_actor IN "Actor-Characteristic"."id_actor"%TYPE,
   id_characteristic IN "Actor-Characteristic"."id_characteristic"%TYPE,
   value IN "Actor-Characteristic"."value_actor_characteristic"%TYPE
)
is
begin
   INSERT INTO "Actor-Characteristic" VALUES(id_actor, id_characteristic,
value);
end;
/

CREATE OR REPLACE procedure actor_characteristic_delete(
   id_actor IN "Actor-Characteristic"."id_actor"%TYPE,
```

```
    id_characteristic IN "Actor-Characteristic"."id_characteristic"%TYPE
)
is
begin
    DELETE FROM "Actor-Characteristic"
    WHERE "id_actor" = id_actor and "id_characteristic" = id_characteristic;
end;
/

CREATE OR REPLACE procedure actor_characteristic_list(
    id_actor IN "Actor-Characteristic"."id_actor"%TYPE,
    list OUT SYS_REFCURSOR
)
is
begin
    open list for
        select "id_characteristic", "type_characteristic" as "характеристика"
        from "Actor-Characteristic" inner join "Characteristic"
using("id_characteristic")
        where "id_actor" = id_actor;
end;
/

CREATE OR REPLACE procedure actor_roles_info(
    id_actor IN "Employee"."id_employee"%TYPE,
    id_gender IN "Employee"."id_gender"%TYPE,
```

```
    age_from IN INT,
    age_to IN INT,
    date_from IN "Repertoire"."performance_date_repertoire"%TYPE,
    date_to IN "Repertoire"."performance_date_repertoire"%TYPE,
    id_genre IN "Genre"."id_genre"%TYPE,
    id_age_category IN "Age_category"."id_age_category"%TYPE,
    id_director IN "Employee"."id_employee"%TYPE,
    actor_roles_cur OUT SYS_REFCURSOR)
is
begin
    open actor_roles_cur for
        select ("name_employee" || ' ' || "surname_employee" || ' ' ||
"middle_name_employee") as "актер",
            "name_role" as "роль", "name_show" as "спектакль", "name_genre" as
"жанр",
            "name_age_category" as "возрастная категория",
            (select "name_employee" || ' ' || "surname_employee" || ' ' ||
"middle_name_employee"
            from "Employee" where "id_employee" = "id_director") as "режиссер-
постановщик"
        from ((("Direction"
            inner join "Employee" on "id_actor" = "id_employee")
            inner join "Role" using ("id_role"))
            inner join "Show" using ("id_show"))
            inner join "Genre" using ("id_genre"))
            inner join "Age_category" using("id_age_category")
        where
```

```
        "id_actor" = NVL(id_actor, "id_actor")
        and "id_gender" = NVL(id_gender, "id_gender")
        and TRUNC(((SELECT SYSDATE FROM DUAL) -
"birthday_employee")) >= NVL(age_from,
                                        TRUNC(((SELECT SYSDATE FROM
DUAL) - "birthday_employee")))
        and TRUNC(((SELECT SYSDATE FROM DUAL) -
"birthday_employee")) <= NVL(age_to,
                                        TRUNC(((SELECT SYSDATE FROM
DUAL) - "birthday_employee")))
        and "id_genre" = NVL(id_genre, "id_genre")
        and "id_age_category" = NVL(id_age_category, "id_age_category")
        and "id_director" = NVL(id_director, "id_director")
        and "id_show" in (select "id_show"
                from "Repertoire"
                where "performance_date_repertoire" >= NVL(date_from,
"performance_date_repertoire")
                    and "performance_date_repertoire" <= NVL(date_to,
"performance_date_repertoire"));


end;
/


CREATE OR REPLACE procedure get_characteristics_list(list OUT
SYS_REFCURSOR)
is
begin
   open list for
```

```
    select "id_characteristic", "type_characteristic"
    from "Characteristic";
end;
/
```

ПРИЛОЖЕНИЕ 5. СКРИПТ С ТЕСТОВЫМ НАБОРОМ ДАННЫХ.

```
INSERT INTO "Job_types" VALUES(0, NULL, 'актер');
/* должно выполниться */

INSERT INTO "Job_types" VALUES(0, NULL, 'постановщик');
/* должно выполниться */

INSERT INTO "Job_types" VALUES(0, 2, 'режиссер-постановщик');
/* должно выполниться */

INSERT INTO "Job_types" VALUES(0, 2, 'художник-постановщик');
/* должно выполниться */

INSERT INTO "Job_types" VALUES(0, 2, 'дирижер-постановщик');
/* должно выполниться */

INSERT INTO "Job_types" VALUES(0, NULL, 'музыкант');
/* должно выполниться */

INSERT INTO "Job_types" VALUES(0, NULL, 'уборщик');
/* должно выполниться */

INSERT INTO "Job_types" VALUES(0, NULL, 'NoName');
/* должно выполниться */

DELETE FROM "Job_types" WHERE "name_job_type" LIKE 'NoName';
/* должно выполниться */


INSERT INTO "Education" VALUES(0, 'среднее');
/* должно выполниться */

INSERT INTO "Education" VALUES(0, 'среднее специальное');
/* должно выполниться */
```

```
INSERT INTO "Education" VALUES(0, 'студент');
/* должно выполниться */

INSERT INTO "Education" VALUES(0, 'высшее');
/* должно выполниться */

INSERT INTO "Education" VALUES(0, 'NoName');
/* должно выполниться */

DELETE FROM "Education" WHERE "name_education" LIKE 'NoName';
/* должно выполниться */


INSERT INTO "Gender" VALUES(0, 'мужской');
/* должно выполниться */

INSERT INTO "Gender" VALUES(0, 'женский');
/* должно выполниться */

INSERT INTO "Gender" VALUES(0, 'NoName');
/* должно выполниться */

DELETE FROM "Gender" WHERE "name_gender" LIKE 'NoName';
/* должно выполниться */


INSERT INTO "Rank" VALUES(0, 'заслуженный артист');
/* должно выполниться */

INSERT INTO "Rank" VALUES(0, 'народный артист');
/* должно выполниться */

INSERT INTO "Rank" VALUES(0, 'лауреат');
/* должно выполниться */

INSERT INTO "Rank" VALUES(0, 'NoName');
/* должно выполниться */

DELETE FROM "Rank" WHERE "name_rank" LIKE 'NoName';
/* должно выполниться */
```

```
INSERT INTO "Competition" VALUES(0, 'Голос');
/* должно выполниться */

INSERT INTO "Competition" VALUES(0, 'Таланты России');
/* должно выполниться */

INSERT INTO "Competition" VALUES(0, 'NoName');
/* должно выполниться */

DELETE FROM "Competition" WHERE "name_competiton" LIKE 'NoName';
/* должно выполниться */


INSERT INTO "Age_category" VALUES(0, 'дети');
/* должно выполниться */

INSERT INTO "Age_category" VALUES(0, 'взрослые');
/* должно выполниться */

INSERT INTO "Age_category" VALUES(0, 'NoName');
/* должно выполниться */

DELETE FROM "Age_category" WHERE "name_age_category" LIKE 'NoName';
/* должно выполниться */


INSERT INTO "Genre" VALUES(0, 'музыкальная комедия');
/* должно выполниться */

INSERT INTO "Genre" VALUES(0, 'трагедия');
/* должно выполниться */

INSERT INTO "Genre" VALUES(0, 'оперетта');
/* должно выполниться */

INSERT INTO "Genre" VALUES(0, 'NoName');
/* должно выполниться */

DELETE FROM "Genre" WHERE "name_genre" LIKE 'NoName';
/* должно выполниться */
```

```
INSERT INTO "Country" VALUES(0, 'Священная Римская империя');
/* должно выполниться */

INSERT INTO "Country" VALUES(0, 'NoName');
/* должно выполниться */

DELETE FROM "Country" WHERE "name_country" LIKE 'NoName';
/* должно выполниться */


INSERT INTO "Author" VALUES(0, 'Людвиг', 'ван Бетховен', '', 18, 1);
/* должно выполниться */

INSERT INTO "Author" VALUES(0, 'Людвиг ван', 'Бетховен', '', 18, 1);
/* должно выполниться */

INSERT INTO "Author" VALUES(0, 'NoName', '', '', 18, 1);
/* должно выполниться */

DELETE FROM "Author" WHERE "name_author" LIKE 'NoName';
/* должно выполниться */


DELETE FROM "Country" WHERE "name_country" LIKE 'Священная Римская
империя';                    /* не должно выполниться */


INSERT INTO "Musical_instruments" VALUES(0, 'гитара');
/* должно выполниться */

INSERT INTO "Musical_instruments" VALUES(0, 'NoName');
/* должно выполниться */

DELETE FROM "Musical_instruments" WHERE "name_instrument" LIKE
'NoName';                        /* должно выполниться */


INSERT INTO "Characteristic" VALUES(0, 'худой');
/* должно выполниться */

INSERT INTO "Characteristic" VALUES(0, 'высокий');
/* должно выполниться */
```

```
INSERT INTO "Characteristic" VALUES(0, 'NoName');
/* должно выполниться */

DELETE FROM "Characteristic" WHERE "type_characteristic" LIKE 'NoName';
/* должно выполниться */


/* тест Employee 1---------------------- ----------------------------------------------------
------------------------------*/

INSERT INTO "Employee" VALUES(0, 'Иван', '', '', 1, TO_DATE('2019/01/01',
'yyyy/mm/dd'), TO_DATE('2018/01/01', 'yyyy/mm/dd'),

   0, 30000, 4, 1);                                                    /* не должно
выполниться */

INSERT INTO "Employee" VALUES(0, 'Иван', '', '', 1, TO_DATE('2021/01/01',
'yyyy/mm/dd'), TO_DATE('2022/01/01', 'yyyy/mm/dd'),

   0, 30000, 4, 1);                                                    /* не должно
выполниться */

INSERT INTO "Employee" VALUES(0, 'Иван', '', '', 1, TO_DATE('1999/01/01',
'yyyy/mm/dd'), TO_DATE('2018/01/01', 'yyyy/mm/dd'),

   0, 30000, 4, 1);                                                     /* должно
выполниться */

UPDATE "Employee" SET "hire_date_employee" = TO_DATE('1998/01/01',
'yyyy/mm/dd')

   WHERE "id_employee" = 1;                                                /* не
должно выполниться */

UPDATE "Employee" SET "birthday_employee" = TO_DATE('2020/01/01',
'yyyy/mm/dd')

   WHERE "id_employee" = 1;                                                /* не
должно выполниться */

UPDATE "Employee" SET "hire_date_employee" = TO_DATE('2020/01/01',
'yyyy/mm/dd'),
```

```
   "birthday_employee" = TO_DATE('1998/01/01', 'yyyy/mm/dd') WHERE
"id_employee" = 1;                /* должно выполниться */

/*-------------------------------------------------------------------------------------------------
-------------------------*/

INSERT INTO "Employee" VALUES(0, 'Оксана', '', '', 2,
TO_DATE('1999/01/01', 'yyyy/mm/dd'), TO_DATE('2020/01/01', 'yyyy/mm/dd'),

   0, 30000, 4, 3);                                                    /* должно
выполниться */

INSERT INTO "Employee" VALUES(0, 'Алексей', '', '', 1,
TO_DATE('1999/01/01', 'yyyy/mm/dd'), TO_DATE('2020/01/01', 'yyyy/mm/dd'),

   0, 30000, 4, 4);                                                    /* должно
выполниться */

INSERT INTO "Employee" VALUES(0, 'Рафаэль', '', '', 1,
TO_DATE('1999/01/01', 'yyyy/mm/dd'), TO_DATE('2020/01/01', 'yyyy/mm/dd'),

   0, 30000, 4, 5);                                                    /* должно
выполниться */

INSERT INTO "Employee" VALUES(0, 'Марсель', '', '', 1,
TO_DATE('1999/01/01', 'yyyy/mm/dd'), TO_DATE('2020/01/01', 'yyyy/mm/dd'),

   0, 30000, 4, 1);                                                    /* должно
выполниться */

INSERT INTO "Employee" VALUES(0, 'Лариса', '', '', 2, TO_DATE('1999/01/01',
'yyyy/mm/dd'), TO_DATE('2020/01/01', 'yyyy/mm/dd'),

   0, 30000, 4, 1);                                                    /* должно
выполниться */

INSERT INTO "Employee" VALUES(0, 'Ксения', '', '', 2,
TO_DATE('1999/01/01', 'yyyy/mm/dd'), TO_DATE('2020/01/01', 'yyyy/mm/dd'),

   0, 30000, 4, 1);                                                    /* должно
выполниться */

INSERT INTO "Employee" VALUES(0, 'Алексей', '', '', 1,
TO_DATE('1999/01/01', 'yyyy/mm/dd'), TO_DATE('2020/01/01', 'yyyy/mm/dd'),
```

```
   0, 30000, 4, 1);                                                    /* должно
выполниться */

INSERT INTO "Employee" VALUES(0, 'NoName', '', '', 1,
TO_DATE('1999/01/01', 'yyyy/mm/dd'), TO_DATE('2020/01/01', 'yyyy/mm/dd'),

   0, 30000, 4, 1);                                                    /* должно
выполниться */

INSERT INTO "Employee" VALUES(0, 'Владимир', '', '', 2,
TO_DATE('1999/01/01', 'yyyy/mm/dd'), TO_DATE('2020/01/01', 'yyyy/mm/dd'),

   0, 30000, 1, 6);                                                    /* должно
выполниться */

INSERT INTO "Employee" VALUES(0, 'Максим', '', '', 1,
TO_DATE('1999/01/01', 'yyyy/mm/dd'), TO_DATE('2020/01/01', 'yyyy/mm/dd'),

   0, 30000, 4, 6);                                                    /* должно
выполниться */

INSERT INTO "Employee" VALUES(0, 'Юрий', '', '', 1, TO_DATE('1999/01/01',
'yyyy/mm/dd'), TO_DATE('2020/01/01', 'yyyy/mm/dd'),

   0, 30000, 1, 7);                                                    /* должно
выполниться */

INSERT INTO "Employee" VALUES(0, 'Гузель', '', '', 2, TO_DATE('1999/01/01',
'yyyy/mm/dd'), TO_DATE('2020/01/01', 'yyyy/mm/dd'),

   0, 30000, 1, 3);                                                    /* должно
выполниться */

/* тест простейших таблиц ----------------------------------------------------------
----------------------------------*/

DELETE FROM "Job_types" WHERE "name_job_type" LIKE 'актер';
/* не должно выполниться */


DELETE FROM "Education" WHERE "name_education" LIKE 'высшее';
/* не должно выполниться */
```

```
DELETE FROM "Gender" WHERE "name_gender" LIKE 'мужской';
/* не должно выполниться */

/* тест Actor-Rank ----------------------------------------------------------------------------
-----------------------------*/

INSERT INTO "Actor-Rank" VALUES(1, 1, TO_DATE('1997/01/01',
'yyyy/mm/dd'), 2);                    /* не должно выполниться */

INSERT INTO "Actor-Rank" VALUES(1, 1, TO_DATE('2020/01/01',
'yyyy/mm/dd'), 2);                     /* должно выполниться */

UPDATE "Actor-Rank" SET "obtaining_date_actor_rank" =
TO_DATE('1997/01/01', 'yyyy/mm/dd')

   WHERE "id_actor" = 1;                                                    /* не
должно выполниться */                                                  /* не
должно выполниться */

UPDATE "Actor-Rank" SET "obtaining_date_actor_rank" =
TO_DATE('2019/01/01', 'yyyy/mm/dd')

   WHERE "id_actor" = 1;                                                     /*
должно выполниться */

/* тест простейших таблиц ---------------------------------------------------------------
---------------------------------*/

DELETE FROM "Rank" WHERE "name_rank" LIKE 'заслуженный артист';
/* не должно выполниться */


DELETE FROM "Competition" WHERE "name_competiton" LIKE 'Таланты
России';                        /* не должно выполниться */

/* тест Show 1 ----------------------------------------------------------------------------
-----------------------------*/

INSERT INTO "Show" VALUES(0, 'Творения Прометея', 2, 3, 4, 1, 1, 2, 17,

   TO_DATE('2022/01/01', 'yyyy/mm/dd'));
/* не должно выполниться */

INSERT INTO "Show" VALUES(0, 'Творения Прометея', 2, 3, 4, 1, 1, 2, 19,
```

```
   TO_DATE('2020/01/01', 'yyyy/mm/dd'));
/* не должно выполниться */

INSERT INTO "Show" VALUES(0, 'Творения Прометея', 2, 3, 4, 1, 1, 2, 19,

   TO_DATE('2022/01/01', 'yyyy/mm/dd'));
/* должно выполниться */

UPDATE "Show" SET "century_show" = 17 WHERE "id_show" = 1;
/* не должно выполниться */

UPDATE "Show" SET "premier_date_show" = TO_DATE('2020/01/01',
'yyyy/mm/dd') WHERE "id_show" = 1;     /* не должно выполниться */

UPDATE "Show" SET "century_show" = 18, "premier_date_show" =
TO_DATE('2025/01/01', 'yyyy/mm/dd')

   WHERE "id_show" = 1;                                                    /*
должно выполниться */

/* тест простейших таблиц ----------------------------------------------------------
----------------------------------*/

DELETE FROM "Genre" WHERE "name_genre" LIKE 'музыкальная комедия';
/* не должно выполниться */


DELETE FROM "Age_category" WHERE "name_age_category" LIKE
'взрослые';                          /* не должно выполниться */


DELETE FROM "Author" WHERE "name_author" LIKE 'Людвиг';
/* не должно выполниться */

UPDATE "Author" SET "life_century_author" = 21 WHERE "id_author" = 1;
/* не должно выполниться */

UPDATE "Author" SET "life_century_author" = 17 WHERE "id_author" = 1;
/* должно выполниться */

/*-------------------------------------------------------------------------------------------
-------------------------*/
```

```
INSERT INTO "Role" VALUES(0, 1, 'Прометей', 1);
/* должно выполниться */

INSERT INTO "Role" VALUES(0, 1, 'Амфион', 0);
/* должно выполниться */

INSERT INTO "Role" VALUES(0, 1, 'None', 0);
/* должно выполниться */

DELETE FROM "Role" WHERE "name_role" LIKE 'None';
/* должно выполниться */


INSERT INTO "Role-Characteristic" VALUES(1, 1, 0);
/* должно выполниться */

INSERT INTO "Role-Characteristic" VALUES(2, 2, 0);
/* должно выполниться */


INSERT INTO "Musician-Instrument" VALUES(11, 1);
/* должно выполниться */

INSERT INTO "Musician-Show" VALUES(11, 1);
/* должно выполниться */

INSERT INTO "Musician-Show" VALUES(10, 1);
/* должно выполниться */

DELETE FROM "Musician-Show" WHERE "id_musician" = 10;
/* должно выполниться */


INSERT INTO "Actor-Characteristic" VALUES(1, 1, 0);
/* должно выполниться */

/* тест Direction 1----------------------------------------------------------------------------
----------------------------*/

INSERT INTO "Direction" VALUES(8, 1, 1);
/* должно выполниться */
```

```
INSERT INTO "Direction" VALUES(8, 2, 0);
/* не должно выполниться */

INSERT INTO "Direction" VALUES(7, 2, 1);
/* должно выполниться */

INSERT INTO "Direction" VALUES(6, 2, 0);
/* не должно выполниться */

INSERT INTO "Direction" VALUES(1, 1, 1);
/* не должно выполниться */

INSERT INTO "Direction" VALUES(1, 1, 0);
/* должно выполниться */

INSERT INTO "Direction" VALUES(5, 1, 1);
/* не должно выполниться */

INSERT INTO "Direction" VALUES(5, 1, 0);
/* не должно выполниться */

/* тест простейших таблиц ----------------------------------------------------------------
----------------------------------*/

DELETE FROM "Role" WHERE "name_role" LIKE 'Прометей';
/* не должно выполниться */


INSERT INTO "Role-Characteristic" VALUES(2, 1, 0);
/* не должно выполниться */

UPDATE "Role-Characteristic" SET "id_characteristic" = 2 WHERE "id_role" =
1;                  /* не должно выполниться */

DELETE FROM "Role-Characteristic" WHERE "id_role" = 1;
/* не должно выполниться */


DELETE FROM "Characteristic" WHERE "type_characteristic" LIKE 'худой';
/* не должно выполниться */

DELETE FROM "Musical_instruments" WHERE "name_instrument" LIKE
'гитара';                  /* не должно выполниться */
```

```
/*тест Employee 2-------------------------------------------------------------------------------
------------------------------*/

INSERT INTO "Employee" VALUES(0, 'NoName2', '', '', 1,
TO_DATE('1999/01/01', 'yyyy/mm/dd'), TO_DATE('2018/01/01', 'yyyy/mm/dd'),

   0, 30000, 4, 1);                                                    /* должно
выполниться */

DELETE FROM "Employee" WHERE "name_employee" LIKE 'NoName2';
/* должно выполниться */

DELETE FROM "Employee" WHERE "id_employee" = 1;
/* не должно выполниться */

DELETE FROM "Employee" WHERE "id_employee" = 2;
/* не должно выполниться */

DELETE FROM "Employee" WHERE "id_employee" = 3;
/* не должно выполниться */

DELETE FROM "Employee" WHERE "id_employee" = 4;
/* не должно выполниться */

DELETE FROM "Employee" WHERE "id_employee" = 11;
/* не должно выполниться */

INSERT INTO "Employee" VALUES(0, 'NoName2', '', '', 1,
TO_DATE('1999/01/01', 'yyyy/mm/dd'), TO_DATE('2018/01/01', 'yyyy/mm/dd'),

   0, 30000, 4, 3);                                                    /* должно
выполниться */

DELETE FROM "Employee" WHERE "name_employee" LIKE 'NoName2';
/* должно выполниться */

INSERT INTO "Employee" VALUES(0, 'NoName2', '', '', 1,
TO_DATE('1999/01/01', 'yyyy/mm/dd'), TO_DATE('2018/01/01', 'yyyy/mm/dd'),

   0, 30000, 4, 4);                                                    /* должно
выполниться */

DELETE FROM "Employee" WHERE "name_employee" LIKE 'NoName2';
/* должно выполниться */
```

```
INSERT INTO "Employee" VALUES(0, 'NoName2', '', '', 1,
TO_DATE('1999/01/01', 'yyyy/mm/dd'), TO_DATE('2018/01/01', 'yyyy/mm/dd'),
   0, 30000, 4, 5);                                                    /* должно
выполниться */

DELETE FROM "Employee" WHERE "name_employee" LIKE 'NoName2';
/* должно выполниться */

/*-------------------------------------------------------------------------------------------
-------------------------*/

INSERT INTO "Show" VALUES(0, 'Леонора', 2, 3, 4, 2, 2, 2, 19,
TO_DATE('2022/01/01', 'yyyy/mm/dd'));     /* должно выполниться */

INSERT INTO "Role" VALUES(0, 2, 'Леонора', 1);
/* должно выполниться */

INSERT INTO "Role" VALUES(0, 2, 'Пицарро', 0);
/* должно выполниться */

INSERT INTO "Musician-Show" VALUES(11, 2);
/* должно выполниться */

/* тест Repertoire 1-----------------------------------------------------------------------
-----------------------------*/

INSERT INTO "Repertoire" VALUES(0, 2, TO_DATE('2023/06/01',
'yyyy/mm/dd'));                    /* не должно выполниться */

INSERT INTO "Direction" VALUES(8, 4, 1);
/* должно выполниться */

INSERT INTO "Repertoire" VALUES(0, 2, TO_DATE('2023/06/01',
'yyyy/mm/dd'));                    /* не должно выполниться */

INSERT INTO "Direction" VALUES(6, 4, 0);
/* должно выполниться */

INSERT INTO "Repertoire" VALUES(0, 2, TO_DATE('2023/06/01',
'yyyy/mm/dd'));                    /* не должно выполниться */

INSERT INTO "Direction" VALUES(7, 5, 1);
/* должно выполниться */
```

```
INSERT INTO "Repertoire" VALUES(0, 2, TO_DATE('2018/01/01',
'yyyy/mm/dd'));                    /* не должно выполниться */

INSERT INTO "Repertoire" VALUES(0, 2, TO_DATE('2022/01/03',
'yyyy/mm/dd'));                    /* должно выполниться */

INSERT INTO "Repertoire" VALUES(0, 2, TO_DATE('2022/01/03',
'yyyy/mm/dd'));                    /* не должно выполниться */

INSERT INTO "Repertoire" VALUES(0, 2, TO_DATE('2023/01/01',
'yyyy/mm/dd'));                    /* должно выполниться */

INSERT INTO "Repertoire" VALUES(0, 2, TO_DATE('2024/01/01',
'yyyy/mm/dd'));                    /* должно выполниться */

DELETE FROM "Repertoire" WHERE "id_performance" = 3;
/* должно выполниться */

/* тест Ticket ----------------------------------------------------------------------------
----------------------------*/

INSERT INTO "Ticket" VALUES(0, 1, 10, 100, 0, NULL);
/* должно выполниться */

INSERT INTO "Ticket" VALUES(0, 1, 10, 100, 0, NULL);
/* не должно выполниться */

INSERT INTO "Ticket" VALUES(0, 1, 11, 100, 0, NULL);
/* должно выполниться */

/* тест Repertoire 2----------------------------------------------------------------------
----------------------------*/

DELETE FROM "Repertoire" WHERE "id_performance" = 1;
/* не должно выполниться */

/* тест Show 2----------------------------------------------------------------------------
----------------------------*/

UPDATE "Show" SET "premier_date_show" = TO_DATE('2030/01/01',
'yyyy/mm/dd') WHERE "id_show" = 2;     /* не должно выполниться */

UPDATE "Show" SET "premier_date_show" = TO_DATE('2005/01/01',
'yyyy/mm/dd') WHERE "id_show" = 2;     /* не должно выполниться */
```

```
DELETE FROM "Show" WHERE "id_show" = 2;
/* не должно выполниться */

/*-------------------------------------------------------------------------------------------------
-------------------------*/

INSERT INTO "Show" VALUES(0, 'NoName', 2, 3, 4, 1, 1, 2, 19,
TO_DATE('2022/01/01', 'yyyy/mm/dd'));      /* должно выполниться */

INSERT INTO "Role" VALUES(0, 3, 'NoName', 1);
/* должно выполниться */

INSERT INTO "Direction" VALUES(8, 6, 1);
/* должно выполниться */

INSERT INTO "Direction" VALUES(9, 6, 0);
/* должно выполниться */

/* тест Direction 2-------------------------------------------------------------------------------
-----------------------------*/

DELETE FROM "Direction" WHERE "id_actor" = 9 and "id_role" = 6;
/* должно выполниться */

INSERT INTO "Direction" VALUES(9, 6, 0);
/* должно выполниться */

INSERT INTO "Repertoire" VALUES(0, 3, TO_DATE('2024/01/01',
'yyyy/mm/dd'));                          /* должно выполниться */

UPDATE "Show" SET "century_show" = 21 WHERE "id_show" = 3;
/* не должно выполниться */

INSERT INTO "Role" VALUES(0, 3, 'NoName2', 1);
/* должно выполниться */

INSERT INTO "Direction" VALUES(1, 7, 1);
/* не должно выполниться */

DELETE FROM "Direction" WHERE "id_actor" = 9 and "id_role" = 6;
/* не должно выполниться */

/* тест Tour ---------------------------------------------------------------------------------
----------------------------*/
```

```
INSERT INTO "Tour" VALUES(12, 1, TO_DATE('2021/01/01', 'yyyy/mm/dd'),
   TO_DATE('2022/01/01', 'yyyy/mm/dd'), 1);
/* не должно выполниться */

INSERT INTO "Tour" VALUES(1, 2, TO_DATE('2021/01/01', 'yyyy/mm/dd'),
   TO_DATE('2022/01/01', 'yyyy/mm/dd'), 1);
/* не должно выполниться */

INSERT INTO "Tour" VALUES(1, 1, TO_DATE('2023/01/01', 'yyyy/mm/dd'),
   TO_DATE('2021/01/01', 'yyyy/mm/dd'), 1);
/* не должно выполниться */

INSERT INTO "Tour" VALUES(13, 1, TO_DATE('2021/01/01', 'yyyy/mm/dd'),
   TO_DATE('2022/01/01', 'yyyy/mm/dd'), 1);
/* не должно выполниться */

INSERT INTO "Tour" VALUES(1, 1, TO_DATE('2020/01/01', 'yyyy/mm/dd'),
   TO_DATE('2021/01/01', 'yyyy/mm/dd'), 1);
/* не должно выполниться */

INSERT INTO "Tour" VALUES(1, 1, TO_DATE('2022/01/01', 'yyyy/mm/dd'),
   TO_DATE('2023/01/01', 'yyyy/mm/dd'), 1);
/* должно выполниться */

INSERT INTO "Tour" VALUES(2, 1, TO_DATE('2022/01/01', 'yyyy/mm/dd'),
   TO_DATE('2023/01/01', 'yyyy/mm/dd'), 1);
/* должно выполниться */

INSERT INTO "Tour" VALUES(3, 1, TO_DATE('2022/01/01', 'yyyy/mm/dd'),
   TO_DATE('2023/01/01', 'yyyy/mm/dd'), 1);
/* должно выполниться */

INSERT INTO "Tour" VALUES(4, 1, TO_DATE('2022/01/01', 'yyyy/mm/dd'),
   TO_DATE('2023/01/01', 'yyyy/mm/dd'), 1);
/* должно выполниться */

INSERT INTO "Tour" VALUES(2, 2, TO_DATE('2022/03/01', 'yyyy/mm/dd'),
```

```
   TO_DATE('2024/01/01', 'yyyy/mm/dd'), 1);
/* не должно выполниться */

INSERT INTO "Tour" VALUES(2, 2, TO_DATE('2023/03/01', 'yyyy/mm/dd'),
   TO_DATE('2024/01/01', 'yyyy/mm/dd'), 1);
/* должно выполниться */

/* тест Subscription ------------------------------------------------------------------------------
-----------------------------*/

INSERT INTO "Subscription" VALUES(0, 1, 1);
/* не должно выполниться */

INSERT INTO "Subscription" VALUES(0, 1, NULL);
/* должно выполниться */

INSERT INTO "Subscription" VALUES(0, NULL, 1);
/* должно выполниться */

UPDATE "Subscription" SET "id_genre" = 1 WHERE "id_subscription" = 2;
/* не должно выполниться */

UPDATE "Subscription" SET "id_author" = 1 WHERE "id_subscription" = 1;
/* не должно выполниться */

UPDATE "Subscription" SET "id_genre" = 2 WHERE "id_subscription" = 1;
/* должно выполниться */

UPDATE "Subscription" SET "id_author" = 2 WHERE "id_subscription" = 2;
/* должно выполниться */

INSERT INTO "Subscription" VALUES(0, NULL, 1);
/* должно выполниться */

/* тест Ticket-Subscription -----------------------------------------------------------------------
-------------------------------*/

INSERT INTO "Ticket-Subscription" VALUES(1, 1);
/* должно выполниться */

INSERT INTO "Ticket-Subscription" VALUES(1, 2);
/* не должно выполниться */
```

```
INSERT INTO "Ticket-Subscription" VALUES(2, 3);
/* не должно выполниться */

/* тест Musician-Show ------------------------------------------------------------------------
--------------------------------*/

INSERT INTO "Musician-Show" VALUES(10, 2);
/* не должно выполниться */

DELETE FROM "Musician-Show" WHERE "id_musician" = 11;
/* не должно выполниться */

/*-------------------------------------------------------------------------------------------
-------------------------*/
```