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*.*** 3](#_Toc40140848)

[**ГЛАВА 1. АНАЛИЗ ПРОЕКТА.** 3](#_Toc40140849)

[**ГЛАВА 2. ПРОЕКТИРОВАНИЕ СИСТЕМЫ.** 6](#_Toc40140850)

[**ГЛАВА 3. РЕАЛИЗАЦИЯ СИСТЕМЫ.** 12](#_Toc40140851)

[**ГЛАВА 4. ТЕСТИРОВАНИЕ.** 19](#_Toc40140852)

[**ЗАКЛЮЧЕНИЕ.** 20](#_Toc40140853)

[**ЛИТЕРАТУРА.** 22](#_Toc40140854)

[**ПРИЛОЖЕНИЕ 1. СКРИПТ СОЗДАНИЯ СХЕМЫ БД.** 22](#_Toc40140855)

[**ПРИЛОЖЕНИЕ 2. СКРИПТ УДАЛЕНИЯ СХЕМЫ БД.** 41](#_Toc40140856)

[**ПРИЛОЖЕНИЕ 3. СКРИПТЫ СОЗДАНИЯ ТРИГГЕРОВ ДЛЯ ОГРАНИЧЕНИЯ ЦЕЛОСТНОСТИ БД.** 45](#_Toc40140857)

[**ПРИЛОЖЕНИЕ 4. СОЗДАНИЕ ХРАНИМЫХ ПРОЦЕДУР.** 74](#_Toc40140858)

[**ПРИЛОЖЕНИЕ 5. СКРИПТ С ТЕСТОВЫМ НАБОРОМ ДАННЫХ.** 101](#_Toc40140859)

# **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.

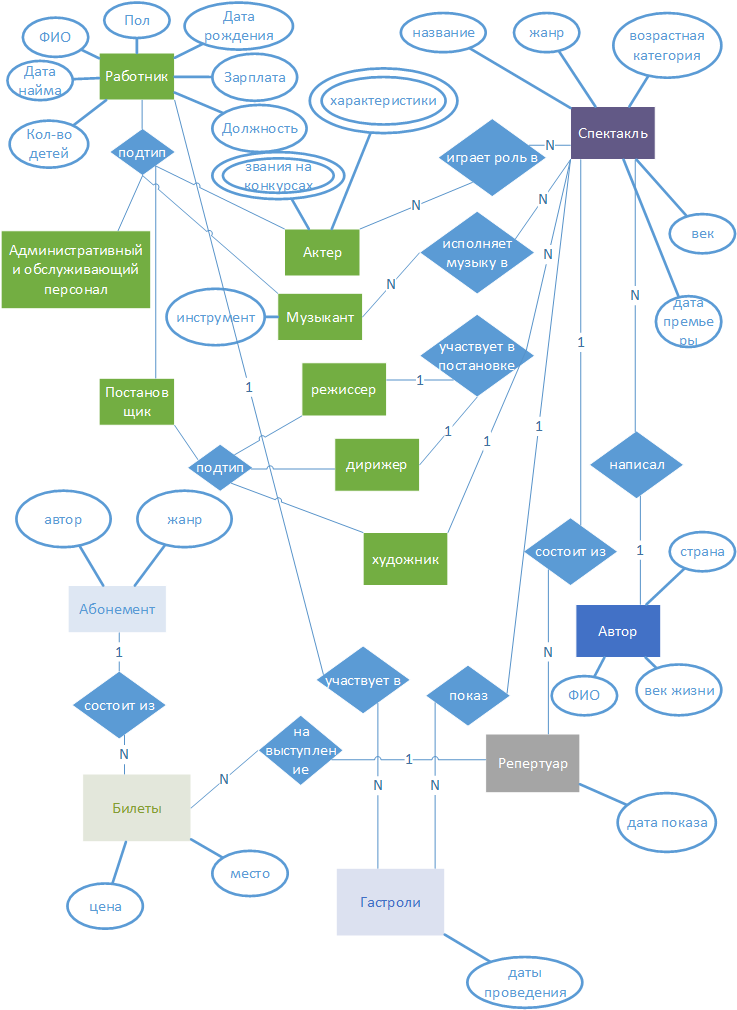
***Задачи проектного задания***:

* 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.



***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.

* 1. в качестве сервера выступает машина с установленной БД Oracle;
  2. в качестве клиента выступает любая машина, поддерживающая виртуальную машину Java и имеющая соединение с сетью Интернет.
  3. Алгоритм взаимодействия клиента и сервера:

1. клиентское приложение связывается по известному IP-адресу с сервером посредством JDBC.
2. клиентское приложение проводит аутентификацию пользователя;
3. клиентское приложение предоставляет пользователю формы, необходимые его роли;
4. клиентское приложение совершает вызовы хранимых процедур БД посредством JDBC в зависимости от задач пользователя;
5. 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).

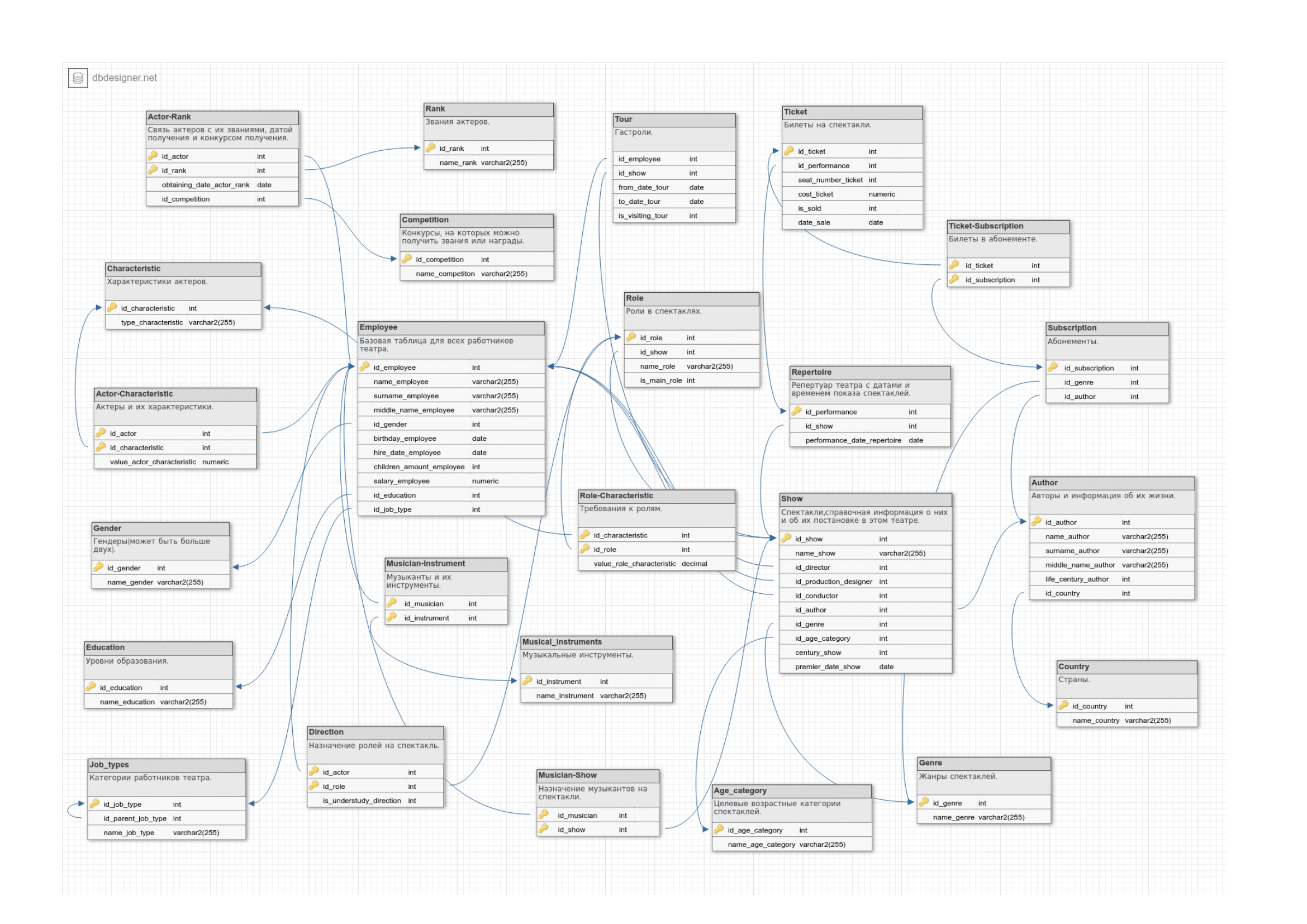
***Способ представления супертипов и подтипов:***

1. иерархия работников:
   * 1. супертип работника представлен таблицей Employee;
     2. 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.
2. 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 cо схемой БД: 



***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:
  + cкрипт 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:***

* cкрипт 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");

* cкрипт 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:***

* cкрипт 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;

* cкрипт 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 "диpижеp-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 \*/

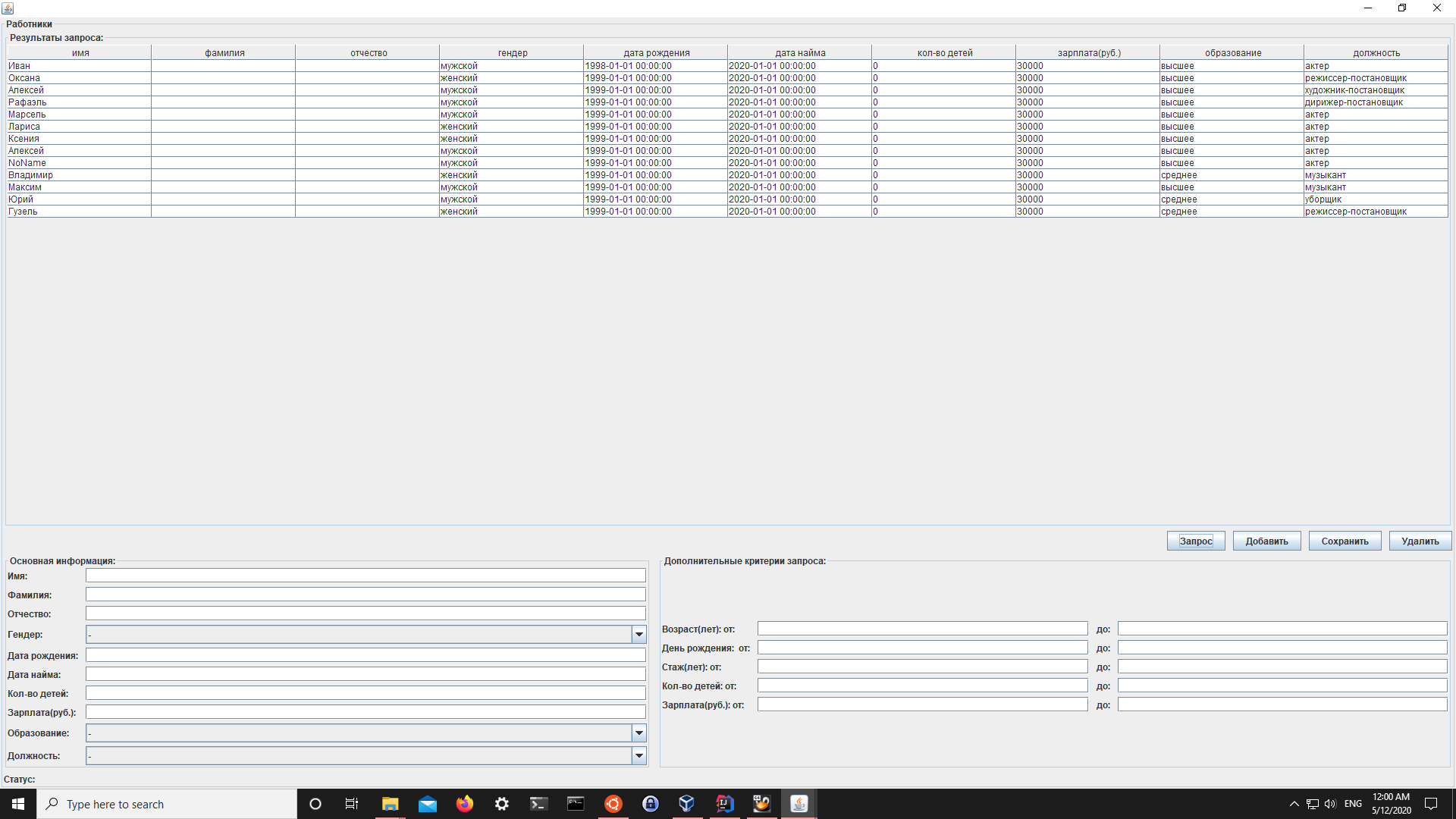
INSERT INTO "Author" VALUES(0, 'NoName', '', '', 18, 1); /\* должно выполниться \*/

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 c использованием встроенного в Intellij IDEA редактора форм.
* пример формы:



Обработка ошибок и исключений происходит на клиентской стороне. При получении ошибки отображается JOptionPane c 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:***
   1. Описаны основные сущности и отношения, определяемые проектным заданием, выделены группы сущностей, определено наличие отношений супертип - подтип, приведена ER диаграмма;
   2. Выявлены требования к обеспечению целостности данных, и основных путях обеспечения их выполнения;
   3. Выявлены основные роли пользователей приложения и основные сценарии использования их взаимодействия с приложением. Приведена диаграмма прецедентов;
   4. Выполнено полно и качественно.
2. ***Проведено проектирование системы.***
   1. Приведена общая архитектура приложения и ее основные части с описанием алгоритмов их взаимодействия;
   2. 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;
   3. Построены алгоритмы обеспечения целостности данных, определено разделение ответственности за обеспечение целостности между ядром СУБД, слоем хранимых процедур и алгоритмами клиентской части приложения;
   4. 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;
   5. Решены вопросы авторизации и ее разнесения на уровни СУБД и клиентской части приложения;
   6. Выполнено полно и качественно.
3. ***Система реализована.***
   1. Описан общий объем работ по программированию (в соответствии с ранее описанной архитектурой;
   2. Описана номенклатура SQL скриптов для построения схемы данных с иллюстрированием фрагментами кода;
   3. Приведены характеристики тестового набора данных и скрипты SQL для ввода тестового набора данных в систему;
   4. Описана номенклатура PL/SQL скриптов, обеспечивающих различные аспекты целостности данных с иллюстрированием фрагментами кода;
   5. Описана реализация пользовательского интерфейса системы;
   6. Освещены вопросы обработки ошибок и исключений с иллюстрированием программным кодом;
   7. Выполнено полно и качественно.
4. ***Система протестирована.***
   1. Приведены сведения о критериях для тестового набора данных;
   2. Перечислен порядок работ по тестированию приложения;
   3. Описаны результаты проведенного тестирования в связи с проектным заданием;
   4. Выполнено полно и качественно.

Поставленная цель полностью достигнута.

# **ЛИТЕРАТУРА.**

1. [ER-модель, Нотация П. Чена](https://ru.wikipedia.org/wiki/ER-%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C)
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 != 'pежиссеp-постановщик'

and employee\_job\_type != 'художник-постановщик'

and employee\_job\_type != 'диpижеp-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 "диpижеp-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 "диpижеp-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 'pежиссеp-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 'диpижеp-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 'pежиссеp-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, 'pежиссеp-постановщик'); /\* должно выполниться \*/

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

INSERT INTO "Job\_types" VALUES(0, 2, 'диpижеp-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, 'тpагедия'); /\* должно выполниться \*/

INSERT INTO "Genre" VALUES(0, 'опеpетта'); /\* 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 \*/

/\*----------------------------------------------------------------------------------------------------------------------------\*/