МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет

по проектной работе

РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИНФОРМАЦИОННОЙ СИСТЕМЫ ТЕАТРА

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

Новосибирск, 2020

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ.

Цель проектного задания – создание информационной системы театра.

Назначение: использование в театрах.

При анализе проектного задания были выделены следующие бизнес-процессы:

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

Основные группы пользователей системы:

- Посетители театра;
- Работники театра.

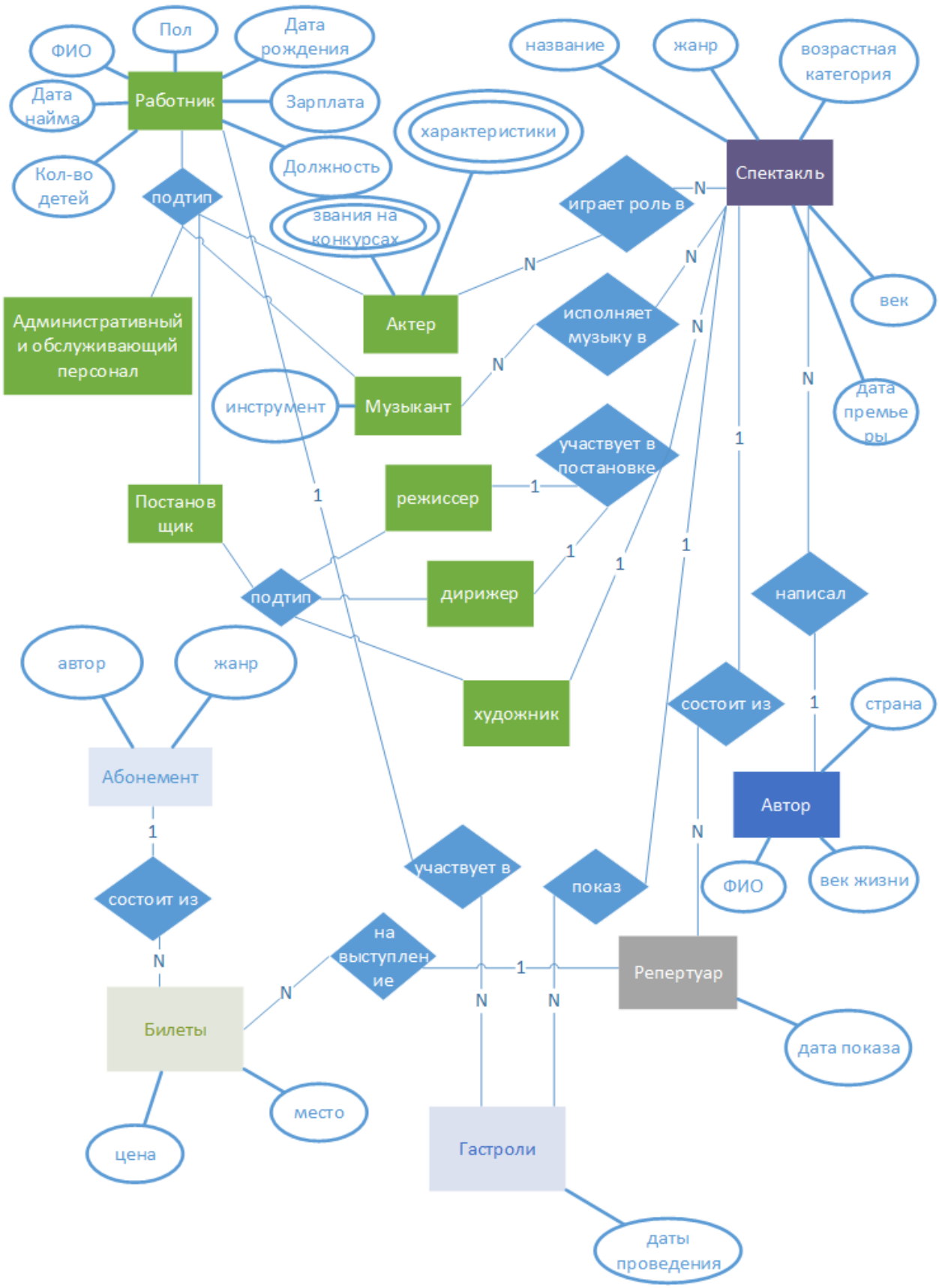
Задачи проектного задания:

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

ГЛАВА 1. АНАЛИЗ ПРОЕКТА.

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

Группа сущностей с отношением супертип-подтип выделена зеленым цветом.


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

Требования к обеспечению целостности данных:

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

Основные пути обеспечения целостности:

- На уровне базы данных:
    - Триггеры вставки, обновления и удаления;

    - Хранимые процедуры;
    - Каскадное удаление данных;
    - Первичные таблицы заполняется исключительно администратором базы данных.
- На уровне клиентского приложения:
    - Информация для ввода выдается выпадающими списками и только та, которая необходима;
    - Проверка вводимых данных на соответствие шаблону средствами языка;
    - Различные требования на заполнение полей;
    - Отсутствуют элементы интерфейса, позволяющие пользователю как-либо навредить целостности базы данных.

Роли для разрабатываемого приложения:

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

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

Кассир	Продажа билетов и абонементов	13	Пользователь	-

ГЛАВА 2. ПРОЕКТИРОВАНИЕ СИСТЕМЫ.

Архитектура приложения: клиент-сервер.

a. В качестве сервера выступает машина с установленной БД Oracle;
b. В качестве клиента выступает любая машина, поддерживающая виртуальную машину Java и имеющая соединение с сетью Интернет.

Алгоритм взаимодействия клиента и сервера:

a. Клиентское приложение связывается по известному IP-адресу с сервером посредством JDBC.
b. Клиентское приложение проводит аутентификацию пользователя;
c. Клиентское приложение предоставляет пользователю формы, необходимые его роли;
d. Клиентское приложение совершает вызовы хранимых процедур БД посредством JDBC в зависимости от задач пользователя;
e. Клиентское приложение завешает сеанс работы с сервером.

Основные таблицы и группы таблиц:

- Таблицы с характеристиками и свойствами:
  - Characteristic (характеристики актеров);
  - Gender (пол);
  - Education (уровни образования);
  - Job_types (должности(профессии));
  - Age_category (возрастные категории);
  - Genre (жанры спектаклей);
  - Country (страны);
  - Rank (звания);
  - Competition (конкурсы, на которых можно получить звания);
  - Musical_instruments (музыкальные инструменты).
- Основные таблицы, реализующие сущности:
  - Employee (работники);
  - Show (спектакли);
  - Role (роли спектаклей);
  - Repertoire (репертуар);

- Author (авторы);
- Tour (гастроли);
- Ticket (билеты);
- Subscription (Абонементы).

- Таблицы, реализующие отношения между сущностями и атрибуты этих сущностей:
  - Actor-Rank (актеры и их звания);
  - Actor-Characteristic (характеристики актеров);
  - Direction (назначение актеров на роли);
  - Musician-Show (назначение музыкантов на спектакли);
  - Musician-instrument (музыканты и их инструменты);
  - Role-Characteristic (требования к актеру для назначения на роль);
  - Ticket- Subscription (состав абонемента из билетов).
- Аутентификация пользователей:
  - Users (таблица с информацией о пользователях системы, их ролях);
  - User-Role (таблица с ролями пользователей системы).

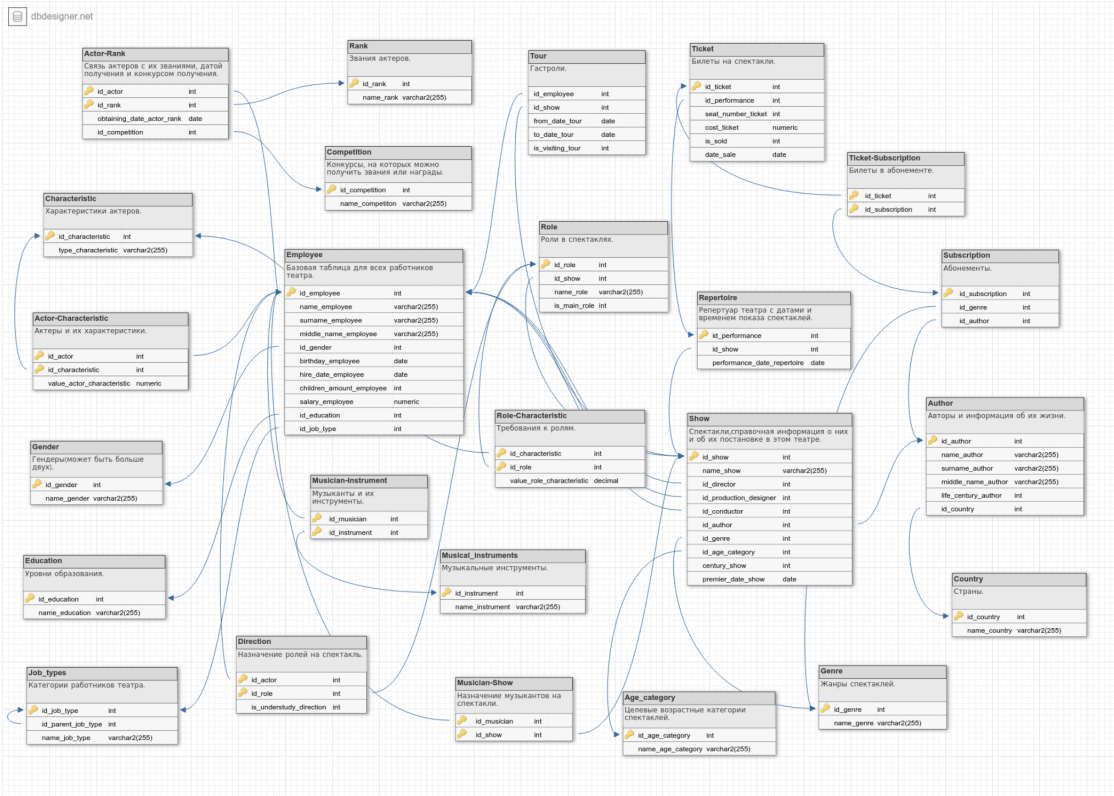
Способ представления супертипов и подтипов:

a. Иерархия работников:
   i. Супертип работника представлен таблицей Employee;
   ii. Подтипы определяются полем профессии в таблице Employee из таблицы Job_types и набором триггеров, которые реагируют на подтипы в зависимости от значения в этом поле.

b. Иерархия типов профессий:

   В таблице Job_types 3 поля: id_job_type – номер профессии, id_parent_job_type – номер родительской профессии и name_job_type – название профессии. Тип профессии высший в иерархии имеет в поле id_parent_job_type значение Null, а его подтипы в поле id_parent_job_type имеют id_job_type этой профессии.

Диаграмма схемы БД:



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
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:
  - 14 форм на языке программирования Java;
- Серверная часть на языках SQL и PL\SQL:
  - Скрипт создания **27** таблиц в СУБД;
  - Скрипт создания **42** триггеров в СУБД;
  - Скрипт создания **86** хранимых процедур в СУБД;
  - Скрипт удаления ограничений целостности, последовательностей, триггеров, таблиц.

При написании приложения использовался **Oracle Database 11.2.0.4 JDBC Thin Driver (ojdbc6)** для связи с серверной стороной**.**

В самом клиентском приложении запросы не формируются, клиентское приложение только вызывает сохраненные на сервере процедуры и функции. Если необходимо, то получает данные посредством курсора.

Пример подготовки к вызову хранимой процедуры:

```
ticketInfo = connection.prepareCall("{call get_ticket(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)}");
ticketInfo.registerOutParameter(13, OracleTypes.CURSOR);
```

Пример вызова хранимой процедуры и получения возврата из курсора:

```
ticketInfo.setNull(1, OracleTypes.DATE);
ticketInfo.setNull(2, OracleTypes.DATE);
ticketInfo.setNull(3, OracleTypes.INTEGER);
ticketInfo.setNull(4, OracleTypes.INTEGER);
ticketInfo.setNull(5, OracleTypes.INTEGER);
ticketInfo.setNull(6, OracleTypes.INTEGER);
ticketInfo.setNull(7, OracleTypes.INTEGER);
ticketInfo.setNull(8, OracleTypes.INTEGER);
ticketInfo.setNull(9, OracleTypes.INTEGER);
ticketInfo.setNull(10, OracleTypes.INTEGER);
ticketInfo.setNull(11, OracleTypes.INTEGER);
ticketInfo.setNull(12, OracleTypes.INTEGER);
ticketInfo.execute();

ResultSet results = (ResultSet) ticketInfo.getObject(13);
fillTableFromResultSet(ticketsTable, 2, tickets, results);
results.close();
```

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

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

Пример – создание таблицы с работниками и последовательности для ее триггера вставки:

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

CREATE sequence "EMPLOYEE_ID_EMPLOYEE_SEQ";
/

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

- Скрипт theatre_oracle_drop.sql удаляет ограничения целостности, последовательности, триггеры, таблицы.

Пример – удаление одного из представленных в БД объектов и очистка корзины:

```
DROP TRIGGER "ACTOR-RANK-INSERT-UPDATE";
DROP PROCEDURE REPERTOIRE_INFO;
```

```
ALTER TABLE "Employee"
  DROP CONSTRAINT "Employee_fk0";
DROP TABLE "Rank";
DROP SEQUENCE "RANK_ID_RANK_SEQ";
PURGE RECYCLEBIN;
COMMIT;
```

Номенклатура PL\SQL-скриптов для обеспечения целостности данных:

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

Пример - связывание актера с его достижениями. Если вдруг пользователь случайно введет дату получения звания, которая раньше дня рождения этого актера, то система отреагирует исключением:

```
CREATE OR REPLACE trigger "ACTOR-RANK-INSERT-UPDATE"
  before insert or update
  on "Actor-Rank"
  for each row
declare
  birthday "Employee"."birthday_employee"%TYPE;
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;
/
```

- Скрипт theatre_oracle_stored_procedures.sql заводит в БД хранимые процедуры и функции. Часть параметров может быть NULL, используется функция NVL, чтобы не учитывать этот параметр при запросе (сделать условие с ним - тождественно истинным).

Пример – продажа абонемента – это продажа входящих в него билетов:

```
CREATE OR REPLACE procedure sell_subscription(id_subscription IN
"Subscription"."id_subscription"%TYPE)
  is
  today_date DATE;
```

```
  cursor tickets is
    select "id_ticket"
    from "Ticket-Subscription"
    where "id_subscription" = id_subscription;
begin
  select CURRENT_DATE into today_date from dual;

  for ticket in tickets
    loop
      UPDATE "Ticket" SET "is_sold" = 1, "date_sale" = today_date WHERE "id_ticket" =
ticket."id_ticket";
    end loop;

  UPDATE "Subscription" SET "is_sold" = 1 WHERE "id_subscription" = id_subscription;
end;
/
```

Характеристики тестового набора данных:

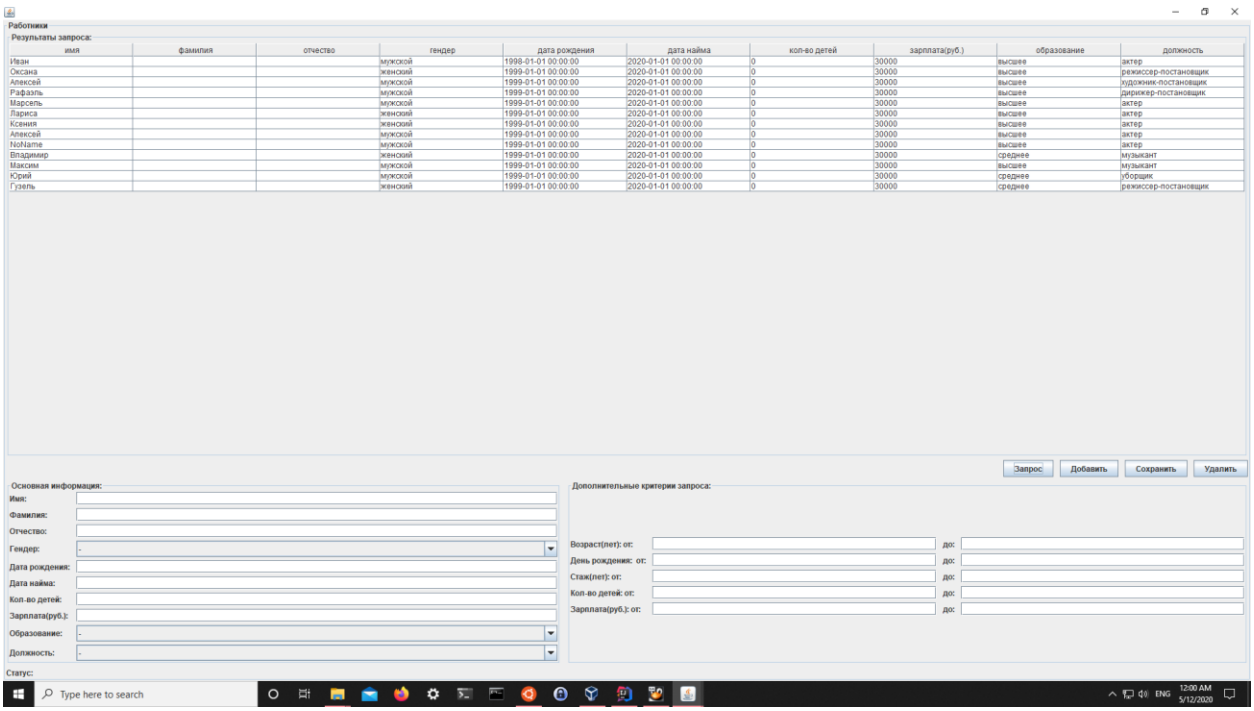
- тестируют правильность заполнения таблиц;
- тестируют правильность работы триггеров, ограничивающих целостность БД.

Пример из скрипт theatre_oracle_test.sql осуществляет тестовое заполнение таблицы Employee:

```
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 */
```

Реализация пользовательского интерфейса системы:

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



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

Пример – удаление автора из БД:

```
try {
   if (resultTable.getSelectedRow() == -1) {
      JOptionPane.showMessageDialog(mainPanel, "Выберите запись для удаления!",
            "Ошибка удаления!", JOptionPane.ERROR_MESSAGE);
      return;
   }
   int selectedRow = resultTable.getSelectedRow();

   deleteAuthor.setInt(1, authors.get(selectedRow));
   deleteAuthor.execute();

   updateResultTable();
} catch (Exception exception) {
   JOptionPane.showMessageDialog(mainPanel, exception.getMessage().split("\n", 2)[0],
         "Ошибка удаления!", 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.

Порядок работ по тестированию приложения:

- проводится проверка правильности заполнения данными или удаления из БД данных с учетом триггеров, срабатывающих при нарушении целостности: вводятся данные из тестового набора данных и проверяется реакция на эти данные, например:

```
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 */
```

  должно завершиться исключением триггера вставки в таблицу Employee, о чем явно сказано в комментарии;
  На этом этапе ошибок выявлено не было.
- проводится проверка правильности возврата хранимыми процедурами и функциями результатов на основе данных, которыми была заполнена БД на 1 этапе: для проверки правильности работы были добавлены работники театра посредством пользовательского интерфейса, из них сформирован спектакль, назначены выступления и выпущены для них билеты. Эти билеты были проданы, была проанализирована статистика продаж. Через формы получения информации была проверена вся информация о добавленных работниках, спектаклях и т. д.
  На этом этапе ошибок выявлено не было.

Результаты:

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

ЗАКЛЮЧЕНИЕ.

1. ***Проведен анализ проекта:***
    a. Описаны основные сущности и отношения, определяемые проектным заданием, выделены группы сущностей, определено наличие отношений супертип - подтип, приведена ER диаграмма;
    b. Выявлены требования к обеспечению целостности данных, и основных путях обеспечения их выполнения;
    c. Выявлены основные роли пользователей приложения и основные сценарии использования их взаимодействия с приложением. Приведена диаграмма прецедентов;
    d. Выполнено полно и качественно.
2. ***Проведено проектирование системы.***
    a. Приведена общая архитектура приложения и ее основные части с описанием алгоритмов их взаимодействия;
    b. Описаны проектные решения логического уровня, включая основные таблицы и их группы, способ представления в реляционной схеме супертипов и подтипов сущностей с обоснованием выбора, сложных моментов логического проектирования и неочевидных решений. Приведена диаграмма схемы БД;
    c. Построены алгоритмы обеспечения целостности данных, определено разделение ответственности за обеспечение целостности между ядром СУБД, слоем хранимых процедур и алгоритмами клиентской части приложения;
    d. Раскрыто общее строение интерфейса в соответствии с ролями пользователей и прецедентами. Приведена таблица роли - прецеденты - формы с перечнем наиболее принципиальных форм интерфейса с кратким описанием функциональности каждой;
    e. Решены вопросы авторизации и ее разнесения на уровни СУБД и клиентской части приложения;
    f. Выполнено полно и качественно.
3. ***Система реализована.***
    a. Описан общий объем работ по программированию (в соответствии с ранее описанной архитектурой);
    b. Описана номенклатура SQL скриптов для построения схемы данных с иллюстрированием фрагментами кода;

c. Приведены характеристики тестового набора данных и скрипты SQL для ввода тестового набора данных в систему;
d. Описана номенклатура PL/SQL скриптов, обеспечивающих различные аспекты целостности данных с иллюстрированием фрагментами кода;
e. Описана реализация пользовательского интерфейса системы;
f. Освещены вопросы обработки ошибок и исключений с иллюстрированием программным кодом;
g. Выполнено полно и качественно.

4. ***Система протестирована.***
   a. Приведены сведения о критериях для тестового набора данных;
   b. Перечислен порядок работ по тестированию приложения;
   c. Описаны результаты проведенного тестирования в связи с проектным заданием;
   d. Выполнено полно и качественно.

Поставленная цель полностью достигнута.

ЛИТЕРАТУРА.

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

ПРИЛОЖЕНИЕ 1. СКРИПТ СОЗДАНИЯ СХЕМЫ БД.

В связи с большим объемом скрипт можно найти в репозитории проекта:

https://github.com/xp10rd/NSU-FIT/blob/master/the-3rd-year/data-bases/pl-sql/theater_info_system/resources/theatre_oracle_create.sql

ПРИЛОЖЕНИЕ 2. СКРИПТ УДАЛЕНИЯ СХЕМЫ БД.

В связи с большим объемом скрипт можно найти в репозитории проекта:

https://github.com/xp10rd/NSU-FIT/blob/master/the-3rd-year/data-bases/pl-sql/theater_info_system/resources/theatre_oracle_drop.sql

ПРИЛОЖЕНИЕ 3. СКРИПТЫ СОЗДАНИЯ ТРИГГЕРОВ ДЛЯ ОГРАНИЧЕНИЯ ЦЕЛОСТНОСТИ БД.

В связи с большим объемом скрипт можно найти в репозитории проекта:

https://github.com/xp10rd/NSU-FIT/blob/master/the-3rd-year/data-bases/pl-sql/theater_info_system/resources/theatre_oracle_triggers.sql

ПРИЛОЖЕНИЕ 4. СОЗДАНИЕ ХРАНИМЫХ ПРОЦЕДУР.

В связи с большим объемом скрипт можно найти в репозитории проекта:

https://github.com/xp10rd/NSU-FIT/blob/master/the-3rd-year/data-bases/pl-sql/theater_info_system/resources/theatre_oracle_stored_procedures.sql

ПРИЛОЖЕНИЕ 5. СКРИПТ С ТЕСТОВЫМ НАБОРОМ ДАННЫХ.

В связи с большим объемом скрипт можно найти в репозитории проекта:

https://github.com/xp10rd/NSU-FIT/blob/master/the-3rd-year/data-bases/pl-sql/theater_info_system/resources/theatre_oracle_test.sql

ПРИЛОЖЕНИЕ 6. ИСХОДНЫЕ КОДЫ КЛИЕНТСКОЙ ЧАСТИ ПРИЛОЖЕНИЯ.

В связи с большим объемом исходные коды можно найти в репозитории проекта:

https://github.com/xp10rd/NSU-FIT/tree/master/the-3rd-year/data-bases/pl-sql/theater_info_system/src/main/java/ru/nsu

ПРИЛОЖЕНИЕ 7. СХЕМЫ.

Схему базы данных в высоком разрешении, а также ER-модель можно найти в репозитории проекта:

https://github.com/xp10rd/NSU-FIT/tree/master/the-3rd-year/data-bases/pl-sql/theater_info_system/docs

ПРИЛОЖЕНИЕ 8. ПРИЛОЖЕНИЕ.

Последние актуальные версии артефактов находятся в репозитории проекта:

Исполняемый файл(кроссплатформенный с зависимостями и виртуальной машиной): https://github.com/xp10rd/NSU-FIT/blob/master/the-3rd-year/data-bases/pl-sql/theater_info_system/resources/theater_info_system.exe

Jar файл(кроссплатформенный с зависимостями без виртуальной машины):

https://github.com/xp10rd/NSU-FIT/blob/master/the-3rd-year/data-bases/pl-sql/theater_info_system/resources/theater_info_system.jar

ПРИЛОЖЕНИЕ 9. ИНСТРУКЦИЯ ПО УСТАНОВКЕ И РАБОТЕ С ПО.

Запуск приложения:

Клиентское приложение **theater_info_system.exe** содержит в себе необходимую версию виртуальной Java-машины, поэтому наличие установленной версии не обязательно.

Тем не менее, может оказаться полезным иметь на компьютере Java версии 8 для более оперативного разрешения проблем, либо если Вы будете запускать приложение на установленной виртуальной машине с использованием файла **theater_info_system.jar**.

Windows: установка не требуется. Запуск осуществляется двойным щелчком мыши по иконке исполняемого файла **theater_info_system.exe**, который находится в папке **resources/** проекта **(**если конфигурация системы стандартная и способ запуска не изменялся пользователем**).**

При возникновении проблем, первым делом имеет смысл проверить настройки Защитника Windows или Брандмауэра (межсетевого экрана) на предмет блокировки соединений.

Linux: установка не требуется. Запуск зависит от конкретного дистрибутива. Наиболее универсальный способ: в командной строке ввести команду:

java -jar theater_info_system.jar

, где файл **theater_info_system.jar** находится в папке **resources/** проекта. При возникновении проблем, удостоверьтесь, что используете Java версии 8(1.8.0). Для проверки текущей версии Java введите в командной строке

java -version

При отличии от вышеуказанной версии, установите и выберите необходимую версию специфичным для Вашего дистрибутива образом.

Mac OS: работоспособность не проверялась. Следуйте указаниям для Linux и может у Вас что-то получится.

Иные ОС: следуйте указаниям для Linux.

Тестирование приложения:

Вы можете зайти в ИС без регистрации, как Гость. Для этого выберите соответствующую кнопку при запуске приложения.

Для каждой из ролей существуют учетные записи в ИС:

Роль	Логин	Пароль
директор	master	1234
продавец	slave	1234
администратор	admin	1234

Пароль маскируется звездочками.

После ввода, нажмите кнопку «Войти». Вы попадете на форму выбора дальнейших форм для данной роли. Нажатие на кнопки будет открывать формы. На каждой форме Вы можете использовать такие элементы интерфейса, как выпадающие списки, поля для ввода текста или чисел.

Элементы таблиц активны, то есть Вы можете кликать по ним (выбирать строки). Смысл выбора строки зависит от формы. Это может быть выбор записи на удаление или получения более детальной информации, а может быть и то, и другое. Это зависит от того, на какую кнопку Вы нажмете после выбора строки.

На некоторых формах есть статусная строка (в самом низу), с помощью который Вы можете узнать успешность выполнения запроса или количество элементов в выдаче запроса.

По вопросам функционирования и найденным ошибкам писать в ВК или на почту.