МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«Санкт-Петербургский государственный университет аэрокосмического приборостроения»

ФАКУЛЬТЕТ СРЕДНЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

ОТЧЕТ О ПРАКТИКЕ   
ЗАЩИЩЕН С ОЦЕНКОЙ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

РУКОВОДИТЕЛИ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| преподаватели |  |  |  | И.А.Юрьева, И.Л.Рохманько, |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |
|  |  |  |  |  |

|  |
| --- |
| ОТЧЕТ ПО УЧЕБНОЙ ПРАКТИКЕ |
| В СОСТАВЕ ПРОФЕССИОНАЛЬНОГО МОДУЛЯ  ПМ 04 Разработка, администрирование и защита баз данных |
|  |
|  |
|  |

ОТЧЕТ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | С021К |  |  |  | В. Д Панков |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2023

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

на прохождение учебной практики обучающегося по специальности

*вид практики*

09.02.07 «Информационные системы и программирование»

*код и наименование специальности*

1. Фамилия, имя, отчество обучающегося: Панков Василий Дмитриевич
2. Группа:С021 Период практики: с 26.01.23 по 08.02.23 г.
3. Тема задания: получение первичных профессиональных навыков, подготовка к освоению общих и профессиональных компетенций, по профессиональному модулю

ПМ.04 Разработка, администрирование и защита баз данных

*код и наименование профессионального модуля*

1. Вопросы, подлежащие изучению:

* Анализ предметной области «БД Театра»;
* Разработка диаграммы Чена;
* Реализация базы данных в конкретной СУБД;
* Разработка запросов к базе данных средствами языка DML;
* Импорт данных в базу данных.

1. Содержание отчетной документации:
   1. Отчёт, включающий в себя:

* титульный лист;
* индивидуальное задание;
* материалы о выполнении индивидуального задания (содержание определяется цикловой комиссией);
* список использованных источников.
  1. Аттестационный лист.

1. Срок представления отчета зам. декана по УПР « 08» февраля 2023 г.

Руководители практики от факультета СПО

преподаватель И.Л. Рохманько

должность, уч. степень, звание подпись, дата инициалы, фамилия

преподаватель И.А.Юрьева

должность, уч. степень, звание подпись, дата инициалы, фамилия

Задание принял к исполнению:

Обучающийся 26.01.2023

Дата подпись инициалы, фамилия

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc125668918)

[1 Анализ предметной области 5](#_Toc125668919)

# ВВЕДЕНИЕ

Решение широкого круга задач требует умения разрабатывать и создавать базы данных, подбирать соответствующие им системы управления базами данных, разрабатывать прикладные программы поиска и обработки информации.

Задачей учебной практики по разработке баз данных по МДК 04.01 «Технология разработки и защиты баз данных» является закрепление теоретического материала междисциплинарного курса МДК 04.01 и получение практических навыков по созданию и использованию баз данных.

# Анализ предметной области

* Предметная область: «БД Театра».
* Название ПО: «Расписание представлений»
* Описание ПО:

Вы работаете в театре, в который каждый день происходит множество представлений: оперы, балеты, хоровое пение и тому подобное.

С развитием технологий вы понимаете, что пользователям проще смотреть афишу в интернете, поэтому вам необходимо реализовать базу данных, которая обеспечит информационную базу для данного ресурса. Необходимо обеспечить поиск по театральным актёрам, режиссёрам и персонажам.

Учтите, что в вашем театре есть несколько корпусов.

* Анализ ПО «Расписание представлений»

В результате анализа предметной области выделены основные объекты, которые указаны в Таблице 1, и свойства объектов, которые указаны в таблицах с 2 по 11.

Таблица 1 Список объектов

|  |  |
| --- | --- |
| № | Объекты |
| 1. | Вид постановки |
| 2 | Автор |
| 3. | Постановка |
| 4. | Вид персонала |
| 5. | Обслуживающий персонал |
| 6. | Актёр |
| 7. | Режиссёр |
| 8. | Роль |
| 9. | Сцена |
| 10. | Афиша |
| 11. | Актёры на дату |
| 12. | Звание актёра |
| 13. | Звание режиссёра |

Таблица 2 Вид постановки

|  |  |  |
| --- | --- | --- |
| Идентификатор вида | Название вида | Особенности |
|  |  |  |

Таблица 3 Автор

|  |  |
| --- | --- |
| Идентификатор автора | ФИО |
|  |  |

Таблица 4 Постановка

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Идентификатор представления | Название представления | Вид представления | Акты | Длительность |
|  |  |  |  |  |

Таблица 5 Должность персонала

|  |  |  |
| --- | --- | --- |
| Идентификатор должности | Наименование должности | Описание |
|  |  |  |

Таблица 6 Обслуживающий персонал

|  |  |  |
| --- | --- | --- |
| Идентификатор персонала | ФИО | Должность |
|  |  |  |

Таблица 7 Актёр

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор актёра | ФИО | Дата рождения | Биография |
|  |  |  |  |

Таблица 8 Режиссёр

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор режиссёра | ФИО | Дата рождения | Биография |
|  |  |  |  |

Таблица 9 Роль

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор роли | Название роли | Идентификатор постановки | Описание |
|  |  |  |  |

Таблица 10 Сцена

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор сцены | Название сцены | Адрес | Описание |
|  |  |  |  |

Таблица 11 Афиша

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Идентификатор в афиши | Идентификатор постановки | Идентификатор сцены | Дата/время проведения представления | Комментарий |
|  |  |  |  |  |

Таблица 12 Актёры на дату

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор афиши | Идентификатор актёра | Идентификатор роли | Комментарий |
|  |  |  |  |

Таблица 13 Звание актёра

|  |  |
| --- | --- |
| Идентификатор звания актёра | Название звания актёра |
|  |  |

Таблица 14 Звание режиссёра

|  |  |
| --- | --- |
| Идентификатор звания режиссёра | Название звания режиссёра |
|  |  |

* Объектная система.

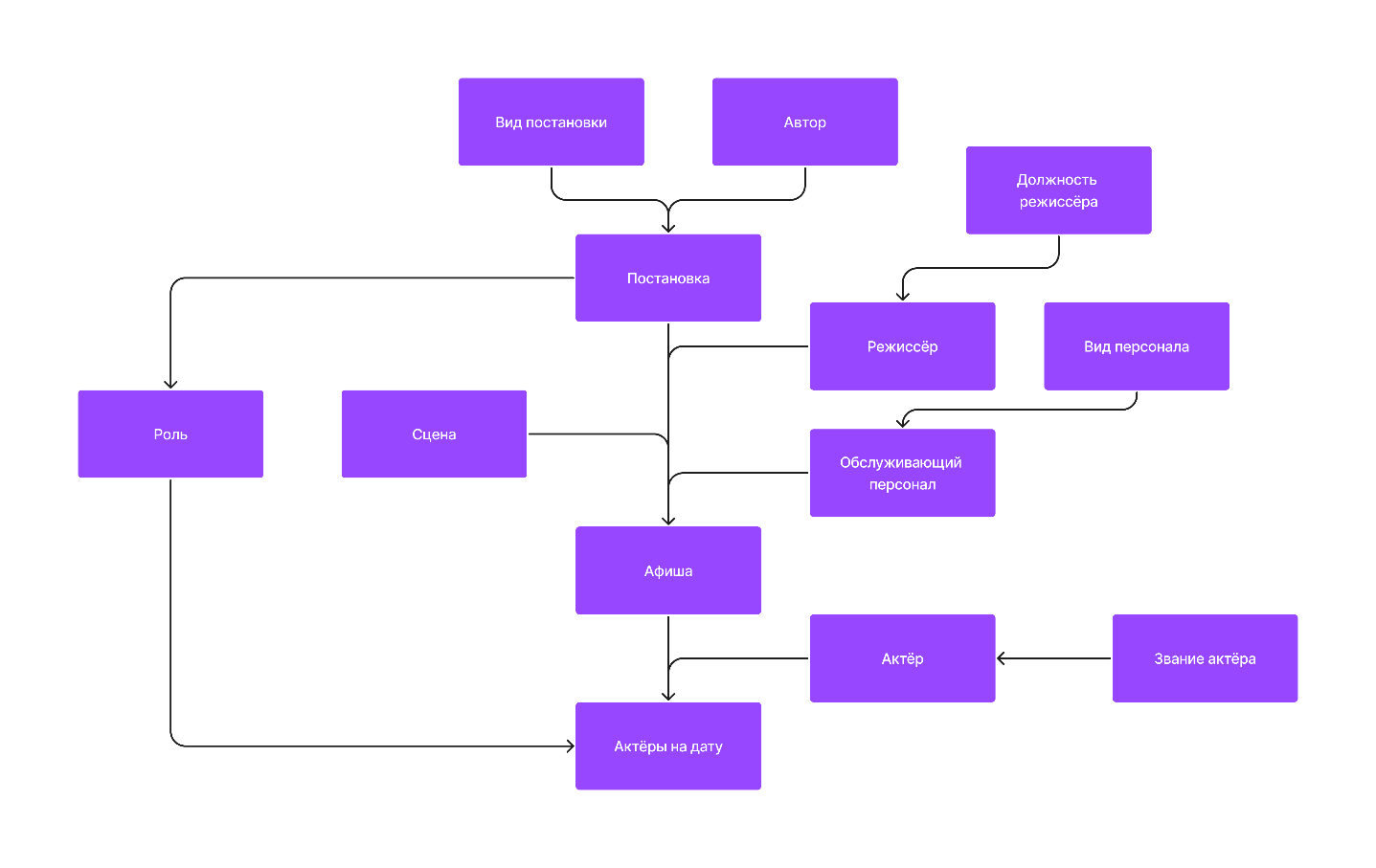


Рисунок 1 Объектная система

# Разработка диаграммы Чена

Диаграммы "сущность-связь" (ERD) предназначены для разработки моделей данных и обеспечивают стандартный способ определения данных и отношений между ними.

Таблица 15 Описание связей

|  |  |
| --- | --- |
| № | Имена связей |
| 1. | Характеризует |
| 2. | Сочиняет |
| 3. | Обслуживает |
| 4. | Выступает |
| 5. | Руководит |
| 6. | Размещает |
| 7. | Выдается |
| 8. | Назначает |
| 9. | Награждает |
| 10. | Участвует |
| 11. | Составляет |
| 12. | Награждает |
| 13. | Распределяет |

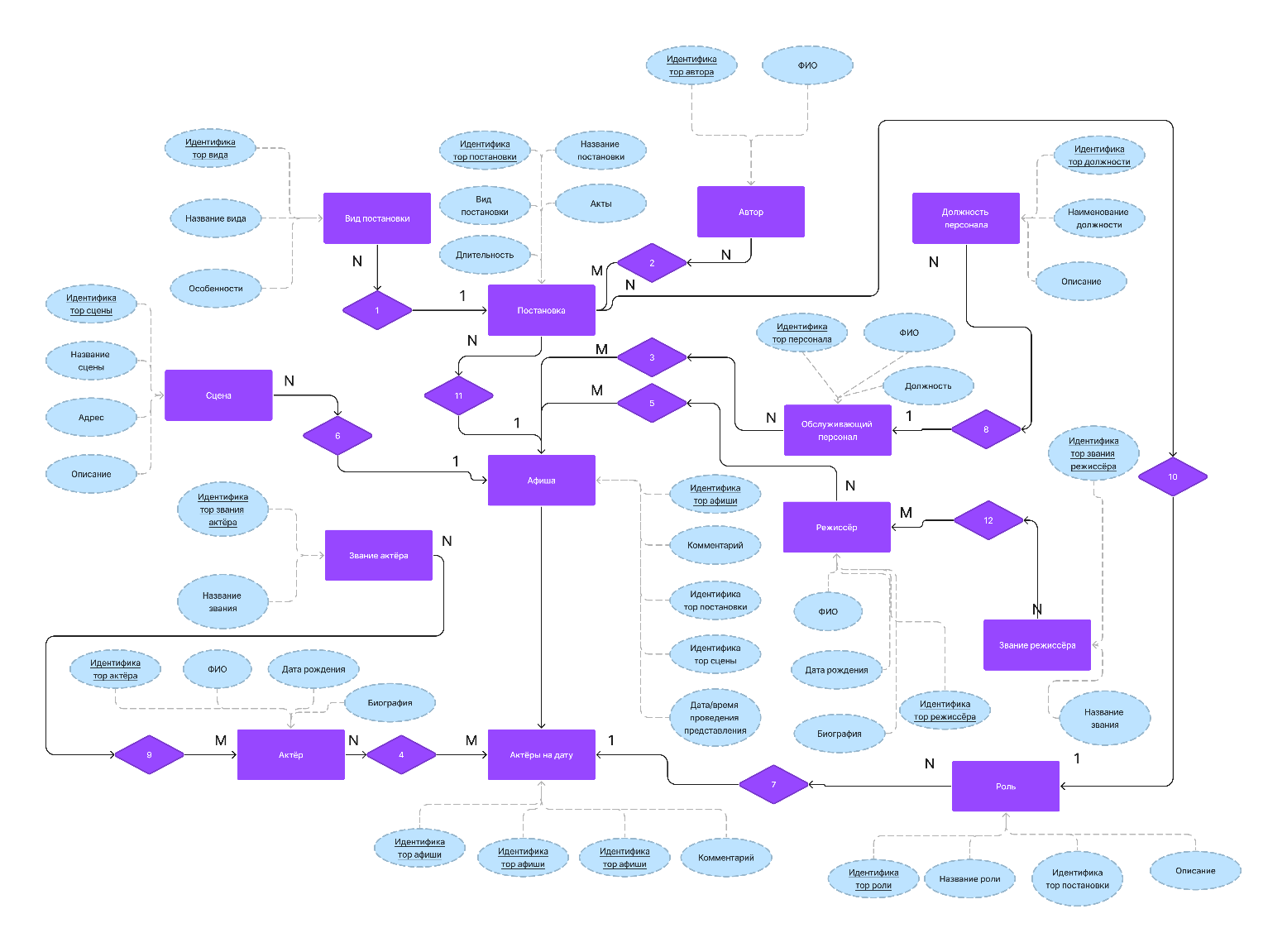


Рисунок 2 Диаграмма Чена

Система создаётся для обслуживания следующих групп пользователей:

* Руководитель театра;
* Бухгалтерия;
* Администратор сайта;
* Пользователь сайта.
  + 1. Функциональные возможности руководителя театра:
       - Редактирование данных о артистах, персонале и режиссёре;
       - Удаление любых данных;
       - Добавление, удаление, редактирование данных о званиях режиссёров и актёров;
    2. Функциональные возможности бухгалтерии:
       - Обновление афиши, а также её редактирование;
       - Редактирование актёров, ролей, авторов и постановок;
       - Добавление новых постановок, видов постановок и авторов;
    3. Функциональные возможности администратора сайта:
       - Чтение всех данных из базы данных;
       - Реализация всех возможных запросов к базе данных;
    4. Функциональные возможности пользователя сайта:
       - Просмотр и поиск информацию об актёрах;
       - Ограниченные возможности фильтрации;
       - Нет возможности редактирования БД;

# 3 Реализация базы данных в конкретной СУБД

# ПРИЛОЖЕНИЕ А SQL- скрипт создания базы данных

Ниже перечислен код для создания таблиц в базе данных:

create table pr\_02.actor

(

id int auto\_increment

primary key,

last\_name varchar(80) not null,

first\_name varchar(80) not null,

second\_name varchar(80) null,

date\_of\_birth date not null,

biography text null

);

create table pr\_02.actor\_rank

(

id int auto\_increment

primary key,

name varchar(80) not null

);

create table pr\_02.actor\_to\_actor\_rank

(

actor\_id int not null,

actor\_rank\_id int not null,

primary key (actor\_id, actor\_rank\_id),

constraint actor\_to\_actor\_rank\_actor\_null\_fk

foreign key (actor\_id) references pr\_02.actor (id)

on update cascade,

constraint actor\_to\_actor\_rank\_actor\_rank\_null\_fk

foreign key (actor\_rank\_id) references pr\_02.actor\_rank (id)

on update cascade

);

create table pr\_02.author

(

id int auto\_increment

primary key,

last\_name varchar(80) not null,

first\_name varchar(80) not null,

second\_name varchar(80) null

);

create table pr\_02.director

(

id int auto\_increment

primary key,

last\_name varchar(80) not null,

first\_name varchar(80) not null,

second\_name varchar(80) null,

date\_of\_birth date not null,

biography text null

);

create table pr\_02.director\_rank

(

id int auto\_increment

primary key,

name varchar(80) not null

);

create table pr\_02.director\_to\_director\_rank

(

director\_id int not null,

director\_rank\_id int not null,

primary key (director\_id, director\_rank\_id),

constraint director\_to\_director\_rank\_director\_null\_fk

foreign key (director\_id) references pr\_02.director (id)

on update cascade,

constraint director\_to\_director\_rank\_director\_rank\_null\_fk

foreign key (director\_rank\_id) references pr\_02.director\_rank (id)

on update cascade

);

create table pr\_02.performance\_type

(

id int auto\_increment

primary key,

name varchar(80) not null,

peculiarities text null

);

create table pr\_02.performance

(

id int auto\_increment

primary key,

name text not null,

type\_id int not null,

acts int not null,

duration int not null comment 'В минутах',

constraint perfomance\_performance\_type\_null\_fk

foreign key (type\_id) references pr\_02.performance\_type (id)

on update cascade,

constraint check\_name

check (`acts` > 0)

);

create table pr\_02.author\_performance

(

author\_id int not null,

performance\_id int not null,

primary key (author\_id, performance\_id),

constraint author\_performance\_author\_null\_fk

foreign key (author\_id) references pr\_02.author (id),

constraint author\_performance\_perfomance\_null\_fk

foreign key (performance\_id) references pr\_02.performance (id)

);

create table pr\_02.role

(

id int auto\_increment

primary key,

name text not null,

performance\_id int not null,

description text null,

constraint role\_performance\_null\_fk

foreign key (performance\_id) references pr\_02.performance (id)

on update cascade

);

create table pr\_02.scene

(

id int auto\_increment

primary key,

name varchar(80) not null,

address text not null,

description int null

);

create table pr\_02.staff\_position

(

id int auto\_increment

primary key,

name varchar(80) not null,

description text not null

)

comment 'Должность персонала';

create table pr\_02.staff

(

id int auto\_increment

primary key,

last\_name varchar(80) not null,

first\_name varchar(80) not null,

second\_name varchar(80) null,

position\_id int not null,

constraint staff\_staff\_position\_null\_fk

foreign key (position\_id) references pr\_02.staff\_position (id)

on update cascade

);

create table pr\_02.timetable

(

id int auto\_increment

primary key,

performance\_id int not null,

scene\_id int not null,

datetime\_start datetime not null,

description int null,

constraint timetable\_pk

unique (performance\_id, scene\_id, datetime\_start),

constraint timetable\_performance\_null\_fk

foreign key (performance\_id) references pr\_02.performance (id)

on update cascade,

constraint timetable\_scene\_null\_fk

foreign key (scene\_id) references pr\_02.scene (id)

on update cascade

)

comment 'Афиша';

create table pr\_02.actor\_to\_date

(

timetable\_id int not null,

actor\_id int not null,

role\_id int not null,

comment text null,

primary key (role\_id, actor\_id, timetable\_id),

constraint actor\_to\_date\_actor\_null\_fk

foreign key (actor\_id) references pr\_02.actor (id)

on update cascade,

constraint actor\_to\_date\_role\_null\_fk

foreign key (role\_id) references pr\_02.role (id)

on update cascade,

constraint actor\_to\_date\_timetable\_null\_fk

foreign key (timetable\_id) references pr\_02.timetable (id)

on update cascade

);

create table pr\_02.director\_to\_timetable

(

director\_id int not null,

timetable\_id int not null,

primary key (director\_id, timetable\_id),

constraint director\_to\_timetable\_director\_null\_fk

foreign key (director\_id) references pr\_02.director (id)

on update cascade,

constraint director\_to\_timetable\_timetable\_null\_fk

foreign key (timetable\_id) references pr\_02.timetable (id)

on update cascade

);

create table pr\_02.staff\_to\_timetable

(

staff\_id int not null,

timetable\_id int not null,

primary key (staff\_id, timetable\_id),

constraint staff\_to\_timetable\_staff\_\_fk

foreign key (staff\_id) references pr\_02.staff (id)

on update cascade,

constraint staff\_to\_timetable\_timetable\_null\_fk

foreign key (timetable\_id) references pr\_02.timetable (id)

on update cascade

);