

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Московский институт электроники и математики им. А.Н. Тихонова

Константинов Алексей Сергеевич, группа БИВ192

Кусакин Илья Константинович, группа БИВ192

Сидоров Иван Владимирович, группа БИВ192

ОТЧЕТ
ПО ДОМАШНЕЙ РАБОТЕ
«Проектирование реляционной базы данных»
предметной области «Пригородные ж/д перевозки»
по дисциплине «Базы данных»

Дата сдачи отчета 24.05.2021

Москва 2021 г.

Оглавление

Оглавление	2
1. Проектирование базы данных	3
1.1 Анализ предметной области	3
1.2 Создание ER-диаграммы	4
1.3 Информационные задачи и круг пользователей БД	5
1.4 Преобразование ER-диаграммы в схему базы данных	6
1.5 Нормализация полученных отношений	9
1.6 Дополнительные ограничения целостности	12
1.7 Описание групп пользователей и прав доступа	12
2. Реализация проекта базы данных	13
2.1 Создание таблиц	13
2.2 Заполнение справочных таблиц	15
2.3 Создание представлений	15
2.4 Назначение прав доступа	18
2.5 Создание триггеров	21
2.6 Создание индексов	26
2.7 Разработка стратегии резервного копирования	27
3. Интерфейс	28
3.1 Используемые технологии	28
3.2 Описание	28
3.3 Демонстрация	30
3.4 Реализация	35

1. Проектирование базы данных

1.1 Анализ предметной области

База данных создается для информационного обслуживания компании, занимающейся пригородными перевозками. БД должна содержать данные о поездах, машинистах, станциях, маршрутах, купленных билетах, рейсах, направлениях и пригородных зонах.

В соответствии с предметной областью система строится с учетом следующих особенностей:

- Каждому направлению соответствует множество станций, но каждая станция относится к одному направлению;
- Каждым направлением заведует один сотрудник, и сотрудник заведует не более чем одним направлением;
- Билет относится к одному сотруднику (кассиру, что его продал), но кассир может относиться к множеству билетов;
- Все станции одного направления имеют уникальные названия;
- Каждый маршрут принадлежит одному направлению, но каждое направление включает в себя множество маршрутов;
- Каждый маршрут включает в себя множество станций, но каждая станция может относиться к множеству маршрутов;
- Каждый рейс соответствует одному маршруту, но по одному маршруту может совершаться множество рейсов;
- Каждая станция относится к множеству билетов и каждый билет относится к нескольким станциям (к двум);
- Каждому рейсу соответствует один машинист, но каждый машинист может совершать множество рейсов;
- Каждому рейсу соответствует один поезд, но каждый поезд может совершать множество рейсов;
- При создании билета указываются две станции: отправления и назначения;
- Маршрут имеет один из режимов движения: ежедневно, по рабочим, по выходным;
- При возникновении изменений в маршруте создается новый маршрут, чтобы сохранить информацию обо всех когда-либо использованных маршрутах;
- Каждая станция относится к одной пригородной зоне, но каждая зона может включать в себя множество станций;
- Стоимость билета рассчитывается как разница между номерами пригородных зон станций отправления и прибытия, умноженная на базовую цену (цену проезда одной пригородной зоны) направления, коэффициент тарифа маршрута и на 2 при выборе услуги “туда-обратно”;
- Стандарт имеет множитель 1, экспресс имеет множитель 2.

1.2 Создание ER-диаграммы

Для создания ER-диаграммы выделим сущности предметной области и определим их атрибуты. Идентифицирующие атрибуты мы выделяем **полужирным** шрифтом, многозначные – *курсивом*, составные подчеркнем.

Сущность «Станция»:

- **Идентификатор**
- Название
- Километр направления
- Пригородная зона

Сущность «Направление»:

- **Название**
- Базовая стоимость

Сущность «Сотрудник»:

- ФИО
- Паспортные данные
- Дата рождения
- Пол
- Должность
- Оклад
- **СНИЛС**
- **ИНН**
- Дата приема на работу
- Дата увольнения

Сущность «Маршрут»:

- **Номер**
- Тариф (стандарт, экспресс)
- Режим движения (ежедневно, по рабочим, по выходным)
- Следование (в город, из города)

Сущность «Поезд»:

- **Идентификатор**
- Модель
- Дата введения в эксплуатацию
- Дата списания

Сущность «Рейс»:

- **Идентификатор**
- Дата

Сущность «Билет»:

- **Идентификатор**
- Стоимость
- Тариф (стандарт, экспресс)
- Дата покупки

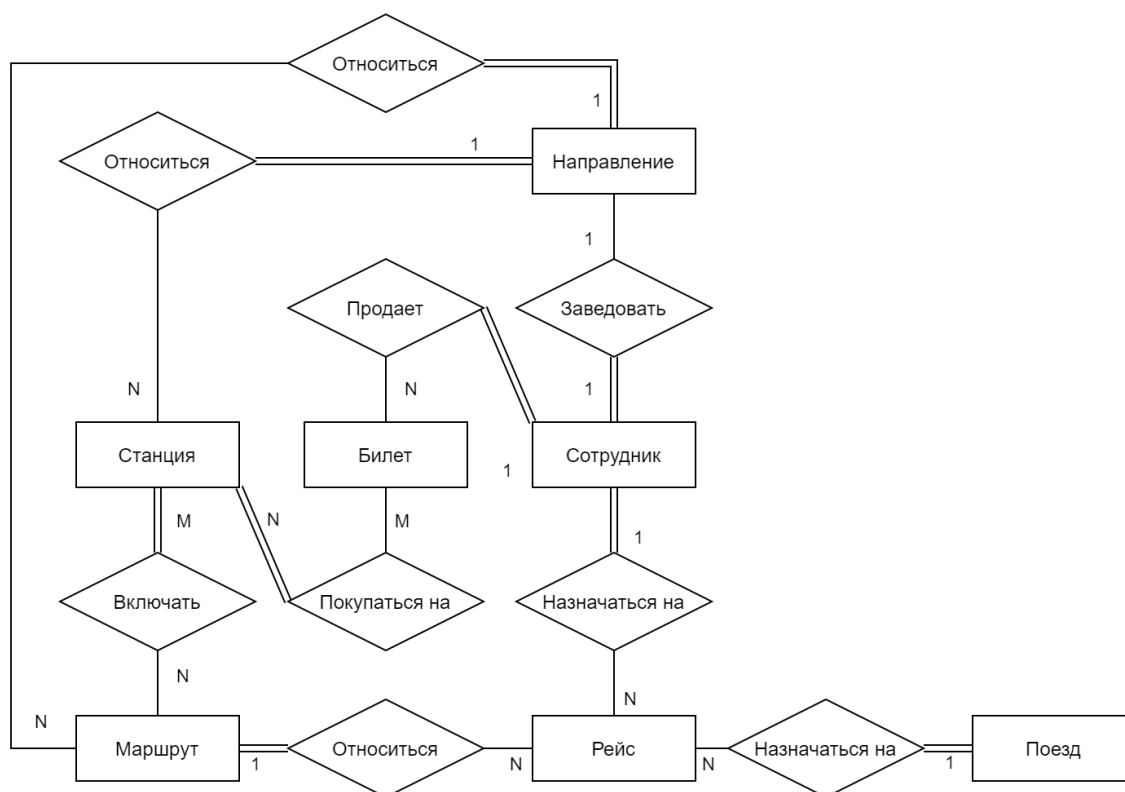


Рис. 1 ER-диаграмма ПрО «Движение пригородных поездов»

1.3 Информационные задачи и круг пользователей БД

- Кассир:
 - Получение полной информации о направлениях
 - Получение полной информации о станциях
 - Добавление строк в таблицу билетов
- Заведующий направлением:
 - Получение информации из всех таблиц
 - Добавление и редактирование информации о маршрутах, рейсах, станциях своего направления
- Менеджер направлений:
 - Получение информации из всех таблиц
 - Добавление и редактирование информации о маршрутах, рейсах, станциях любых направлений
 - Добавление и редактирование направлений
- Менеджер по персоналу:
 - Получение информации о сотрудниках
 - Добавление и обновление сотрудников
- Машинист:
 - Получение информации о своих рейсах и относящихся к ним маршрутах и поездах
- Директор депо:
 - Получение информации о поездах
 - Добавление, редактирование и удаление поездов

1.4 Преобразование ER-диаграммы в схему базы данных

Связь “Покупается на” между сущностями Билет и Станция типа многие-ко-многим, но каждый Билет относится лишь к двум Станциям: отправления и прибытия – поэтому реализуем эту связь через два внешних ключа у отношения Билет.

Связь “Относиться” между отношениями Станция и Маршрут относится к типу многие-ко-многим, для ее разрешения введем вспомогательное отношение “Станции маршрута”, которое содержит комбинации первичных ключей соответствующих исходных отношений.

Связь “Относиться” между Направлением и Станцией относится к типу один-ко-многим, реализуем ему через внешний ключ отношения Станция. Точно так же разрешим связи между отношениями Пригородная зона и Станция (ВнК у отношения Станция), Рейс и Маршрут (ВнК у отношения Рейс), Маршрут и Направление (ВнК у отношения Маршрут), Поезд и Рейс (ВнК у отношения Рейс), Рейс и Сотрудник (ВнК у отношения Рейс), Сотрудник и Направление (ВнК у отношения Направление), Сотрудник и Станция (ВнК у отношения Сотрудник), Сотрудник и Билет (ВнК у отношения Билет).

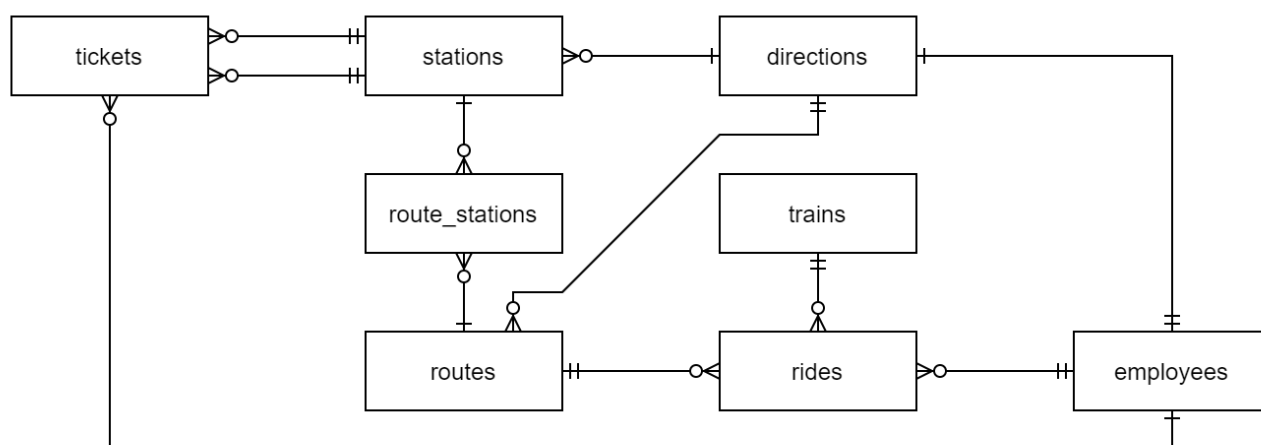


Рис. 2 Схема реляционной базы данных

Табл.1 Направления (directions)

Содержимое поля	Название	Длина	Примечание
Название	name	V(20)	Первичный ключ
Базовая стоимость	dcost	N(6, 2)	Обязательное поле; ≥ 0
Заведующий	manager	V(30)	Внешний ключ

Табл. 2 Станции (stations)

Содержимое поля	Название	Длина	Примечание	
Идентификатор	id	N(6)	Суррогатный первичный ключ	
Название	name	V(20)	Обязательное поле	Уникальны в связке
Направление	direction	V(20)	Внешний ключ	
Пригородная зона	sub_area	N(2)	Обязательное поле	
Километр направления	distance	N(3)	Обязательное поле	

Табл. 3 Маршруты (routes)

Содержимое поля	Название	Длина	Примечание
Идентификатор	id	N(6)	Первичный ключ
Направление	direction	V(20)	Обязательное поле, внешний ключ
Тариф	tariff	V(15)	Обязательное поле; 'стандарт' (по умолчанию) или 'экспресс'
Режим движения	wdays	V(15)	Обязательное поле; 'ежедневно' (по умолчанию), 'по рабочим' или 'по выходным'
Следование	way	V(10)	Обязательное поле; 'в город' (по умолчанию) или 'из города'

Табл. 4 Станции маршрута (route_stations)

Содержимое поля	Название	Длина	Примечание	
Станция	station	N(6)	Внешний ключ	Составной первичный ключ
Маршрут	route	N(6)	Внешний ключ	
Время прибытия	arrive_time	D	Обязательное поле	

Табл. 5 Поезда (trains)

Содержимое поля	Название	Длина	Примечание
Идентификатор	id	N(6)	Первичный ключ
Модель	model	V(20)	Обязательное поле
Дата введения в эксплуатацию	serv_start_date	D	Обязательное поле
Дата списания	serv_end_date	D	Необязательное поле

Табл. 6 Сотрудники (employees)

Содержимое поля	Название	Длина	Примечание
Табельный номер	tabno	V(30)	Первичный ключ
ФИО	full_name	V(40)	Обязательное поле
Паспортные данные	passport	C(10)	Обязательное уникальное поле
Должность	post	V(30)	Обязательное поле
Оклад	salary	N(9, 2)	Обязательное поле, ≥ 12000
Дата рождения	birth_date	D	Обязательное поле
Пол	sex	C(1)	Обязательное поле; 'м' (по умолчанию) или 'ж'
СНИЛС	snils	C(11)	Обязательное уникальное поле
ИНН	inn	C(12)	Обязательное уникальное поле
Дата приема на работу	emp_date	D	Обязательное поле
Дата увольнения	quit_date	D	Необязательное поле

Табл. 7 Рейсы (rides)

<i>Содержимое поля</i>	<i>Название</i>	<i>Длина</i>	<i>Примечание</i>
Идентификатор	id	N(6)	Первичный ключ
Дата	ddate	D	Обязательное поле
Маршрут	route	N(6)	Обязательное поле, внешний ключ
Поезд	train	N(6)	Обязательное поле, внешний ключ
Машинист	machinist	V(30)	Обязательное поле, внешний ключ

Табл. 8 Билеты (tickets)

<i>Содержимое поля</i>	<i>Название</i>	<i>Длина</i>	<i>Примечание</i>
Идентификатор	id	N(6)	Суррогатный первичный ключ
Стоимость	cost	N(6, 2)	Обязательное поле; ≥ 0
Дата покупки	payment_date	D	Обязательное поле
Туда-обратно	round_trip	N(1)	Обязательное поле
Станция отправления	depart_st	N(6)	Обязательное поле, внешний ключ
Станция прибытия	arrive_st	N(6)	Обязательное поле, внешний ключ
Кассир	cashier	V(30)	Внешний ключ

1.5 Нормализация полученных отношений

Создадим справочную таблицу “Тарифы” (tariffs), содержащую информацию о коэффициенте, применяемом к цене билета разного типа.

Создадим справочную таблицу “Модели поездов”.

Создадим справочную таблицу “Должности”.

После нормализации получим следующие таблицы.

Табл. 9 Направления (directions):

Содержимое поля	Название	Длина	Примечание
Название	name	V(20)	Первичный ключ
Базовая стоимость	dcost	N(6, 2)	Обязательное поле; ≥ 0
Заведующий	manager	V(30)	Внешний ключ

Табл. 10 Станции (stations)

Содержимое поля	Название	Длина	Примечание
Идентификатор	id	N(6)	Суррогатный первичный ключ
Название	name	V(20)	Обязательное поле
Направление	direction	V(20)	Внешний ключ
Пригородная зона	sub_area	N(2)	Обязательное поле
Километр направления	distance	N(3)	Обязательное поле

Табл. 11 Тарифы (tariffs)

Содержимое поля	Название	Длина	Примечание
Название	name	V(20)	Первичный ключ
Коэффициент	coef	N(3, 2)	Обязательное поле, > 0

Табл. 12 Маршруты (routes)

Содержимое поля	Название	Длина	Примечание
Идентификатор	id	N(6)	Первичный ключ
Направление	direction	V(20)	Обязательное поле, внешний ключ
Тариф	tariff	V(20)	Обязательное поле, внешний ключ
Режим движения	wdays	V(15)	Обязательное поле; ‘ежедневно’ (по умолчанию), ‘по рабочим’ или ‘по выходным’
Следование	way	V(10)	Обязательное поле; ‘в город’ (по умолчанию) или ‘из города’

Табл. 13 Станции маршрута (route_stations)

Содержимое поля	Название	Длина	Примечание	
Станция	station	N(6)	Внешний ключ	Составной первичный ключ
Маршрут	route	N(6)	Внешний ключ	
Время прибытия	arrive_time	D	Обязательное поле	

Табл. 13 Модели поездов (train_models)

Содержимое поля	Название	Длина	Примечание
Название	model	V(20)	Первичный ключ

Табл. 14 Поезда (trains)

Содержимое поля	Название	Длина	Примечание
Идентификатор	id	N(6)	Первичный ключ
Модель	model	V(20)	Обязательное поле, внешний ключ
Дата введения в эксплуатацию	serv_start_date	D	Обязательное поле
Дата списания	serv_end_date	D	Необязательное поле

Табл. 15 Должности (posts)

Содержимое поля	Название	Длина	Примечание
Название	post	V(30)	Первичный ключ
Оклад	salary	N(9, 2)	Обязательное поле, ≥ 12000

Табл. 16 Сотрудники (employees)

Содержимое поля	Название	Длина	Примечание
Табельный номер	tabno	V(30)	Первичный ключ
Должность	post	V(30)	Обязательное поле, внешний ключ
Фамилия	last_name	V(20)	Обязательное поле
Имя	first_name	V(20)	Обязательное поле
Отчество	patronymic	V(20)	Необязательное поле
Паспортные данные	passport	C(10)	Обязательное уникальное поле
Дата рождения	birth_date	D	Обязательное поле
Пол	sex	C(1)	Обязательное поле; 'м' (по умолчанию) или 'ж'
СНИЛС	snils	C(11)	Обязательное уникальное поле
ИНН	inn	C(12)	Обязательное уникальное поле
Дата приема на работу	emp_date	D	Обязательное поле
Дата увольнения	quit_date	D	Необязательное поле

Табл. 17 Рейсы (rides)

Содержимое поля	Название	Длина	Примечание
Идентификатор	id	N(6)	Первичный ключ
Дата	ddate	D	Обязательное поле
Маршрут	route	N(6)	Обязательное поле, внешний ключ
Поезд	train	N(6)	Обязательное поле, внешний ключ
Машинист	machinist	N(6)	Обязательное поле, внешний ключ

Табл. 18 Билеты (tickets)

Содержимое поля	Название	Длина	Примечание
Идентификатор	id	N(6)	Суррогатный первичный ключ
Стоимость	cost	N(6, 2)	Обязательное поле
Тариф	tariff	V(20)	Обязательное поле, внешний ключ
Дата покупки	payment_date	D	Обязательное поле
Туда-обратно	round_trip	N(1)	Обязательное поле
Станция отправления	depart_st	N(6)	Обязательное поле, внешний ключ
Станция прибытия	arrive_st	N(6)	Обязательное поле, внешний ключ
Кассир	cashier	V(30)	Обязательное поле, внешний ключ

И следующую схему базы данных:

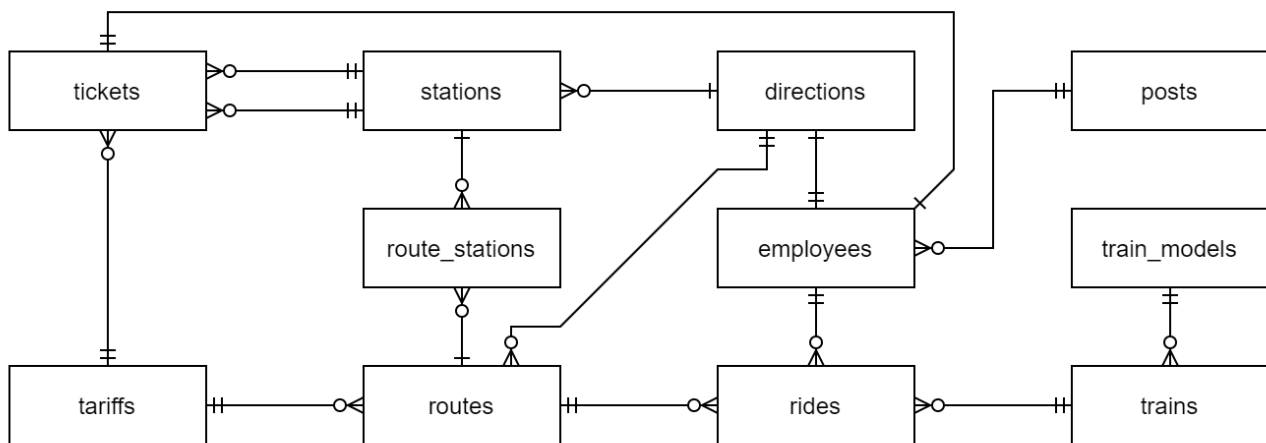


Рис. 3 Схема реляционной базы данных после нормализации

Связь Сотрудники – Билеты 1:М.

Циклы, содержащие таблицу “Тарифы” не создают проблем, так как это справочная таблица – билеты и маршруты через нее не связываются.

Цикл, содержащий таблицу “Сотрудники” не создает проблем, так как согласно ограничениям предметной области на рейсы могут назначаться только сотрудники должности “Машинист”, а заведовать направлениями могут только сотрудники должности “Заведующий направлением”, а также только кассиры могут добавлять билеты.

Цикл “Станции-Маршруты-Направления” разрешается триггером на промежуточную таблицу “Станции маршрута”, проверяющим принадлежность станций маршрута направлению маршрута.

1.6 Дополнительные ограничения целостности

- Поле “Стоимость” таблицы “Билеты” рассчитывается автоматически как разница между номерами пригородных зон станций отправления и прибытия, умноженная на базовую цену направления, коэффициент тарифа маршрута и на 2 при выборе услуги “туда-обратно”;
- Станции отправления и назначения, указанные в билете, должны быть различны и соединены хотя бы одним маршрутом;
- День недели, на который назначается рейс по маршруту, должен соответствовать его режиму следования;
- На рейс могут назначаться только действующие сотрудники должности “Машинист”;
- Заведующими направлениями могут быть назначены только действующие сотрудники должности “Заведующий направлением”;
- На рейсы могут назначаться только не выведенные из эксплуатации поезда;
- Один и тот же поезд не может одновременно выполнять разные рейсы;
- Один и тот же машинист не может одновременно выполнять разные рейсы.

1.7 Описание групп пользователей и прав доступа

Табл. 19 Права доступа к таблицам для групп пользователей

Таблица	Группы пользователей (роли)					
	Кассир	Заведующий направлением	Менеджер направлений	Менеджер по персоналу	Машинист	Директор депо
Станции	R	CRUD	CRUD	—	R	—
Маршруты	R	CRUD	CRUD	—	R	—
Станции маршрута	R	CRUD	CRUD	—	R	—
Направления	R	R	CRUD	—	—	—
Билеты	CRD	R	R	—	—	—
Тарифы	R	R	CRUD	—	—	—
Должности	—	—	—	CRUD	—	—
Сотрудники	R	R	R	CRUD	R	R
Рейсы	R	CRUD	CRUD	R	R	R
Модели поездов	—	—	—	—	R	CRUD
Поезда	—	R	R	—	R	CRUD

2. Реализация проекта базы данных

2.1 Создание таблиц

Отношение posts (Должности):

```
CREATE TABLE posts (  
    post    VARCHAR(30) CONSTRAINT pk_posts PRIMARY KEY,  
    salary  NUMERIC(9, 2) NOT NULL,  
           CONSTRAINT check_salary CHECK (SALARY >= 12000)  
);
```

Отношение employees (Сотрудники):

```
CREATE TABLE employees (  
    tabno    VARCHAR(30) CONSTRAINT pk_employees PRIMARY KEY,  
    post     VARCHAR(30) NOT NULL CONSTRAINT ref_posts REFERENCES posts,  
    last_name VARCHAR(20) NOT NULL,  
    first_name VARCHAR(20) NOT NULL,  
    patronymic VARCHAR(20),  
    passport CHAR(10) UNIQUE NOT NULL,  
    birth_date DATE NOT NULL,  
    sex      CHAR(1) NOT NULL DEFAULT 'М',  
           CONSTRAINT check_sex CHECK (sex IN ('М', 'Ж')),  
    snils     CHAR(11) UNIQUE NOT NULL,  
    inn       CHAR(12) UNIQUE NOT NULL,  
    emp_date  DATE NOT NULL,  
    quit_date DATE  
);
```

Отношение directions (Направления):

```
CREATE TABLE directions (  
    name    VARCHAR(20) CONSTRAINT pk_directions PRIMARY KEY,  
    dcost   NUMERIC(6, 2) NOT NULL,  
           CONSTRAINT check_dcost CHECK (dcost > 0),  
    manager VARCHAR(30) CONSTRAINT ref_manager REFERENCES employees  
);
```

Отношение stations (Станции):

```
CREATE TABLE IF NOT EXISTS stations (  
    id        NUMERIC(6) CONSTRAINT pk_stations PRIMARY KEY,  
    name      VARCHAR(30) NOT NULL,  
    sub_area  NUMERIC(2) NOT NULL,  
    distance  NUMERIC(3) NOT NULL,  
    direction VARCHAR(20) CONSTRAINT ref_direction REFERENCES directions,  
           UNIQUE (name, direction)  
);
```

Отношение tariffs (Тарифы):

```
CREATE TABLE tariffs (  
    name VARCHAR(20) CONSTRAINT pk_tariffs PRIMARY KEY,  
    coef NUMERIC(3, 2) NOT NULL,  
           CONSTRAINT check_coef CHECK (coef > 0)  
);
```

Отношение routes (Маршруты):

```
CREATE TABLE routes (  
  id          NUMERIC(6) CONSTRAINT pk_routes PRIMARY KEY,  
  direction  VARCHAR(20) NOT NULL CONSTRAINT ref_direction REFERENCES directions,  
  tariff     VARCHAR(15) NOT NULL CONSTRAINT ref_route_tariff REFERENCES tariffs,  
  wdays     VARCHAR(15) NOT NULL DEFAULT 'ежедневно',  
             CONSTRAINT check_wdays  
             CHECK (wdays IN ('ежедневно', 'по рабочим', 'по выходным')),  
  way        VARCHAR(10) NOT NULL DEFAULT 'в город',  
             CONSTRAINT check_way  
             CHECK (way IN ('в город', 'из города'))  
);
```

Отношение route_stations (Станции маршрута):

```
CREATE TABLE IF NOT EXISTS route_stations (  
  station     NUMERIC(6) CONSTRAINT ref_rout_st_st REFERENCES stations,  
  route       NUMERIC(6) CONSTRAINT ref_rout_st_route REFERENCES routes,  
  arrive_time TIME WITHOUT TIME ZONE NOT NULL,  
             PRIMARY KEY(station, route)  
);
```

Отношение train_models (Модели поездов):

```
CREATE TABLE train_models (  
  model VARCHAR(20) CONSTRAINT pk_models PRIMARY KEY  
);
```

Отношение trains (Поезда):

```
CREATE TABLE trains (  
  id          NUMERIC(6) CONSTRAINT pk_trains PRIMARY KEY,  
  model       VARCHAR(20) NOT NULL CONSTRAINT ref_models REFERENCES train_models,  
  serv_start_date DATE NOT NULL,  
  serv_end_date DATE  
);
```

Отношение rides (Рейсы):

```
CREATE TABLE rides (  
  id          NUMERIC(6) CONSTRAINT pk_rides PRIMARY KEY,  
  ddate       TIMESTAMP NOT NULL,  
  route       NUMERIC(6) NOT NULL CONSTRAINT ref_route_ride REFERENCES routes,  
  train       NUMERIC(6) NOT NULL CONSTRAINT ref_train REFERENCES trains,  
  machinist   VARCHAR(30) NOT NULL CONSTRAINT ref_machinist REFERENCES employees  
);
```

Отношение tickets (Билеты):

```
CREATE TABLE tickets (  
  id          NUMERIC(6) CONSTRAINT pk_tickets PRIMARY KEY,  
  cost        NUMERIC(6, 2) NOT NULL,  
  tariff      VARCHAR(15) NOT NULL CONSTRAINT ref_route_tariff REFERENCES tariffs,  
  payment_date TIMESTAMP NOT NULL,  
  round_trip  NUMERIC(1) NOT NULL DEFAULT 0,  
             CONSTRAINT check_round_trip CHECK (round_trip IN (0, 1)),  
  depart_st   NUMERIC(6) NOT NULL CONSTRAINT ref_dep_st REFERENCES stations,  
  arrive_st   NUMERIC(6) NOT NULL CONSTRAINT ref_arrive_st REFERENCES stations,  
  cashier     VARCHAR(30) NOT NULL CONSTRAINT ref_cashier REFERENCES employees  
);
```

2.2 Заполнение справочных таблиц

```
INSERT INTO tariffs VALUES
  ('экспресс', 2),
  ('стандарт', 1);

INSERT INTO posts VALUES ('Заведующий направлением', 90000),
  ('Кассир', 40000),
  ('Менеджер по персоналу', 60000),
  ('Машинист', 60000),
  ('Директор депо', 70000),
  ('Менеджер направлений', 80000);
```

2.3 Создание представлений

Активный штат

```
CREATE OR REPLACE VIEW active_staff AS
  SELECT * FROM employees
  WHERE emp_date <= current_date AND
    (quit_date IS NULL OR quit_date IS NOT NULL AND current_date <= quit_date);
```

Машинисты

```
CREATE OR REPLACE VIEW machinists AS
  SELECT * FROM employees
  WHERE post = 'Машинист';
```

Активные машинисты

```
CREATE OR REPLACE VIEW active_machinists AS
  SELECT * FROM active_staff
  WHERE post = 'Машинист';
```

Маршруты своего направления для заведующего

```
CREATE OR REPLACE VIEW manager_routes_verbose AS
  SELECT rt.id, rt.direction, rt.tariff, rt.wdays, rt.way,
    CASE WHEN EXISTS (SELECT * FROM route_stations WHERE route = rt.id) THEN
      (SELECT STRING_AGG(st.name, ', ' ORDER BY
        CASE WHEN rt.way = 'из города' THEN st.distance END ASC,
        CASE WHEN rt.way = 'в город' THEN st.distance END DESC)
      FROM route_stations AS rs, stations AS st
      WHERE rs.station = st.id AND rs.route = rt.id)
    ELSE '-'
  END
  FROM directions AS d, routes AS rt
  WHERE d.manager = user AND rt.direction = d.name
  GROUP BY rt.id, rt.direction, rt.tariff, rt.wdays, rt.way;
ALTER VIEW manager_routes_verbose RENAME COLUMN "case" TO stops;
```

Нагрузка машинистов

```
CREATE OR REPLACE VIEW machinist_workload AS
    SELECT rt.id AS rt_id, emp.last_name || ' ' || emp.first_name || ' ' ||
COALESCE(emp.patronymic, '-') AS machinist, rd.ddate, tr.id AS tr_id
    FROM routes AS rt, employees AS emp, rides AS rd, trains AS tr
    WHERE rd.machinist = emp.tabno AND rd.route = rt.id AND rd.train = tr.id
    ORDER BY rd.ddate
```

Станции направления для заведующего

```
CREATE OR REPLACE VIEW manager_stations AS
    SELECT * FROM stations
    WHERE direction IN (SELECT name FROM directions WHERE manager = user);
```

Расписание

```
CREATE OR REPLACE VIEW rides_verbose AS
    SELECT rides.ddate, rides.train, r.id, r.tariff,
    STRING_AGG(s.name, ', ' ORDER BY
        CASE WHEN r.way = 'из города' THEN s.distance END ASC,
        CASE WHEN r.way = 'в город' THEN s.distance END DESC)
    FROM directions AS d, stations AS s, route_stations AS rs, routes AS r, rides
    WHERE s.direction = d.name AND
        rs.station = s.id AND r.id = rs.route AND r.id = rides.route
    GROUP BY rides.ddate, rides.train, r.id, r.tariff;
ALTER VIEW rides_verbose RENAME COLUMN string_agg TO stops;
```

Будущие рейсы машиниста

```
CREATE OR REPLACE VIEW machinist_rides AS
    SELECT rides.ddate, rides.train, r.id, r.tariff,
    STRING_AGG(s.name, ', ' ORDER BY
        CASE WHEN r.way = 'из города' THEN s.distance END ASC,
        CASE WHEN r.way = 'в город' THEN s.distance END DESC)
    FROM directions AS d, stations AS s,
    route_stations AS rs, routes AS r, rides
    WHERE rides.machinist = user AND s.direction = d.name AND
        rs.station = s.id AND r.id = rs.route AND
        r.id = rides.route AND rides.ddate >= current_timestamp
    GROUP BY rides.ddate, rides.train, r.id, r.tariff;
ALTER VIEW machinist_rides RENAME COLUMN string_agg TO stops;
```

Активные поезда

```
CREATE OR REPLACE VIEW active_trains AS
    SELECT * FROM trains
    WHERE serv_start_date <= current_date AND
        (serv_end_date IS NULL OR serv_end_date IS NOT NULL AND current_date <=
serv_end_date);
```

Статистика билетов по тарифам

```
CREATE OR REPLACE VIEW tickets_stat AS
    SELECT name AS tariff,
        (SELECT COUNT(*) FROM tickets AS tc
        WHERE tc.tariff = tr.name) AS amount,
        (SELECT COALESCE(SUM(tc.cost), 0) FROM tickets AS tc
        WHERE tc.tariff = tr.name) AS total
    FROM tariffs AS tr;
```


Кассиры

```
CREATE OR REPLACE VIEW cashiers AS
SELECT * FROM employees
WHERE post = 'Кассир';
```

Заведующие направлениями

```
CREATE OR REPLACE VIEW route_managers AS
SELECT * FROM employees
WHERE post = 'Заведующий направлением';
```

Табл. 20 Права доступа к представлениям для групп пользователей

Представление	Группы пользователей (роли)					
	Кассир	Заведующий направлением	Менеджер направлений	Менеджер по персоналу	Машинист	Директор депо
Активный штат	—	—	—	CRUD	—	—
Машинисты	—	R	R	CRUD	—	R
Активные машинисты	—	R	R	CRUD	—	R
Маршруты направления	—	CRUD	—	—	—	—
Нагрузка машинистов	—	R	R	R	—	R
Рейсы машиниста	—	—	—	—	R	—
Расписание	R	R	R	—	R	—
Станции направления	—	CRUD	—	—	—	—
Активные поезда	—	R	R	—	R	CRUD
Статистика билетов	—	R	R	R	—	—
Кассиры	R	R	R	CRUD	—	—
Заведующие направлениями	—	R	R	CRUD	—	—

2.4 Назначение прав доступа

Создадим для демонстрации ряд пользователей, соответствующих описанным должностям, и выдадим им соответствующие их должностям права. Имя пользователя соответствует его табельному номеру сотрудника:

- Кассир: e1001
- Заведующий направлением: e201001, e202001 – руководители двух разных направлений
- Менеджер по персоналу: e3001
- Машинист: e4001
- Директор депо: e5001
- Менеджер направлений: e6001

Права доступа к таблицам

```
CREATE USER e1001 WITH PASSWORD '132132';
GRANT SELECT ON stations TO e1001;
GRANT SELECT ON routes TO e1001;
GRANT SELECT ON route_stations TO e1001;
GRANT SELECT ON directions TO e1001;
GRANT SELECT, INSERT, DELETE ON tickets TO e1001;
GRANT SELECT ON tariffs TO e1001;
GRANT SELECT ON posts TO e1001;
GRANT SELECT ON employees TO e1001;
GRANT SELECT ON rides TO e1001;

CREATE USER e201001 WITH PASSWORD '132132';
GRANT SELECT, INSERT, UPDATE, DELETE ON stations TO e201001;
GRANT SELECT, INSERT, UPDATE, DELETE ON routes TO e201001;
GRANT SELECT, INSERT, UPDATE, DELETE ON route_stations TO e201001;
GRANT SELECT ON directions TO e201001;
GRANT SELECT ON tickets TO e201001;
GRANT SELECT ON tariffs TO e201001;
GRANT SELECT ON posts TO e201001;
GRANT SELECT ON employees TO e201001;
GRANT SELECT, INSERT, UPDATE, DELETE ON rides TO e201001;
GRANT SELECT ON train_models TO e201001;
GRANT SELECT ON trains TO e201001;
GRANT SELECT ON route_managers TO e201001;

CREATE USER e202001 WITH PASSWORD '132132';
GRANT SELECT, INSERT, UPDATE, DELETE ON stations TO e202001;
GRANT SELECT, INSERT, UPDATE, DELETE ON routes TO e202001;
GRANT SELECT, INSERT, UPDATE, DELETE ON route_stations TO e202001;
GRANT SELECT ON directions TO e202001;
GRANT SELECT ON tickets TO e202001;
GRANT SELECT ON tariffs TO e202001;
GRANT SELECT ON posts TO e202001;
GRANT SELECT ON employees TO e202001;
GRANT SELECT, INSERT, UPDATE, DELETE ON rides TO e202001;
GRANT SELECT ON train_models TO e202001;
GRANT SELECT ON trains TO e202001;
GRANT SELECT ON route_managers TO e202001;

CREATE USER e3001 WITH PASSWORD '132132';
GRANT SELECT, INSERT, UPDATE, DELETE ON posts TO e3001;
GRANT SELECT, INSERT, UPDATE, DELETE ON employees TO e3001;
GRANT SELECT ON rides TO e3001;
```

```

CREATE USER e4001 WITH PASSWORD '132132';
GRANT SELECT ON stations TO e4001;
GRANT SELECT ON routes TO e4001;
GRANT SELECT ON route_stations TO e4001;
GRANT SELECT ON posts TO e4001;
GRANT SELECT ON employees TO e4001;
GRANT SELECT ON rides TO e4001;
GRANT SELECT ON train_models TO e4001;
GRANT SELECT ON trains TO e4001;

CREATE USER e5001 WITH PASSWORD '132132';
GRANT SELECT ON posts TO e5001;
GRANT SELECT ON employees TO e5001;
GRANT SELECT ON rides TO e5001;
GRANT SELECT, INSERT, UPDATE, DELETE ON train_models TO e5001;
GRANT SELECT, INSERT, UPDATE, DELETE ON trains TO e5001;

CREATE USER e6001 WITH PASSWORD '132132';
GRANT SELECT, INSERT, UPDATE, DELETE ON stations TO e6001;
GRANT SELECT, INSERT, UPDATE, DELETE ON routes TO e6001;
GRANT SELECT, INSERT, UPDATE, DELETE ON route_stations TO e6001;
GRANT SELECT, INSERT, UPDATE, DELETE ON directions TO e6001;
GRANT SELECT ON tickets TO e6001;
GRANT SELECT, INSERT, UPDATE, DELETE ON tariffs TO e6001;
GRANT SELECT ON posts TO e6001;
GRANT SELECT ON employees TO e6001;
GRANT SELECT, INSERT, UPDATE, DELETE ON rides TO e6001;
GRANT SELECT ON train_models TO e6001;
GRANT SELECT ON trains TO e6001;

```

Права доступа к представлениям

```

GRANT SELECT ON cashiers TO e1001;
GRANT SELECT ON rides_verbose TO e1001;

GRANT SELECT ON machinists TO e201001;
GRANT SELECT ON active_machinists TO e201001;
GRANT SELECT, INSERT, UPDATE, DELETE ON manager_routes_verbose TO e201001;
GRANT SELECT ON machinist_workload TO e201001;
GRANT SELECT, INSERT, UPDATE, DELETE ON manager_stations TO e201001;
GRANT SELECT ON active_trains TO e201001;
GRANT SELECT ON tickets_stat TO e201001;
GRANT SELECT ON rides_verbose TO e201001;
GRANT SELECT ON cashiers TO e201001;

GRANT SELECT ON machinists TO e202001;
GRANT SELECT ON active_machinists TO e202001;
GRANT SELECT, INSERT, UPDATE, DELETE ON manager_routes_verbose TO e202001;
GRANT SELECT ON machinist_workload TO e202001;
GRANT SELECT, INSERT, UPDATE, DELETE ON manager_stations TO e202001;
GRANT SELECT ON active_trains TO e202001;
GRANT SELECT ON tickets_stat TO e202001;
GRANT SELECT ON rides_verbose TO e202001;
GRANT SELECT ON cashiers TO e202001;

GRANT SELECT, INSERT, UPDATE, DELETE ON active_staff TO e3001;
GRANT SELECT, INSERT, UPDATE, DELETE ON machinists TO e3001;
GRANT SELECT, INSERT, UPDATE, DELETE ON active_machinists TO e3001;
GRANT SELECT, INSERT, UPDATE, DELETE ON cashiers TO e3001;
GRANT SELECT, INSERT, UPDATE, DELETE ON route_managers TO e3001;
GRANT SELECT ON machinist_workload TO e3001;
GRANT SELECT ON tickets_stat TO e3001;

```

```
GRANT SELECT ON machinist_rides TO e4001;
GRANT SELECT ON active_trains TO e4001;
GRANT SELECT ON rides_verbose TO e4001;

GRANT SELECT ON machinists TO e5001;
GRANT SELECT ON active_machinists TO e5001;
GRANT SELECT ON machinist_workload TO e5001;
GRANT SELECT, INSERT, UPDATE, DELETE ON active_trains TO e5001;

GRANT SELECT ON machinists TO e6001;
GRANT SELECT ON active_machinists TO e6001;
GRANT SELECT ON machinist_workload TO e6001;
GRANT SELECT ON active_trains TO e6001;
GRANT SELECT ON tickets_stat TO e6001;
GRANT SELECT ON route_managers TO e6001;
GRANT SELECT ON rides_verbose TO e6001;
GRANT SELECT ON cashiers TO e6001;
```

2.5 Создание триггеров

Проверка и заполнение билета

```
CREATE OR REPLACE FUNCTION manage_ticket() RETURNS TRIGGER AS $$
DECLARE
    dzone INTEGER;
    st1 RECORD;
    st2 RECORD;
    coef tariffs.coef%TYPE;
    dcost directions.dcost%TYPE;
    total tickets.cost%TYPE;
BEGIN
    IF new.depart_st = new.arrive_st THEN
        RAISE EXCEPTION 'Не удалось добавить билет: станции отправления и назначения
совпадают.';
    END IF;
    SELECT direction, sub_area INTO st1 FROM stations WHERE id = new.depart_st;
    SELECT direction, sub_area INTO st2 FROM stations WHERE id = new.arrive_st;
    IF st1.direction <> st2.direction THEN
        RAISE EXCEPTION 'Не удалось добавить билет: станции разных направлений.';
    END IF;
    IF NOT EXISTS (SELECT * FROM route_stations AS rs1
                    WHERE station = new.arrive_st AND
                        EXISTS (SELECT * FROM route_stations AS rs2 WHERE
                                rs1.route = rs2.route AND
                                rs2.station = new.depart_st AND
                                (rs1.arrive_time > rs2.arrive_time OR
                                 (rs1.arrive_time > '16:00'
                                  AND rs2.arrive_time < '8:00')
                                )
                        )
                    ) THEN
        RAISE EXCEPTION 'Не удалось добавить билет: станции не соединены ни одним
маршрутом.';
    END IF;
    SELECT directions.dcost INTO dcost FROM directions WHERE name = st1.direction;
    SELECT tariffs.coef INTO STRICT coef FROM tariffs WHERE name = new.tariff;

    dzone = ABS(st1.sub_area - st2.sub_area);
    total = dcost * coef * (new.round_trip + 1);
    IF dzone > 1 THEN
        total = total * dzone;
    END IF;
    new.cost = total;
    new.payment_date = current_timestamp;
    new.cashier = user;
    RETURN new;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE EXCEPTION 'Недопустимый тариф: %', new.tariff;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER manage_ticket_tg
    BEFORE INSERT OR UPDATE ON tickets
    FOR EACH ROW
    EXECUTE PROCEDURE manage_ticket();
```

Редактирование представления “Маршруты направления”

```
CREATE OR REPLACE FUNCTION edit_routes() RETURNS TRIGGER AS $$
BEGIN
    IF TG_OP = 'INSERT' THEN
        INSERT INTO routes VALUES (new.id, new.direction, new.tariff, new.wdays, new.way);
    ELSE
        UPDATE routes SET id = new.id, direction = new.direction, tariff = new.tariff,
            wdays = new.wdays, way = new.way WHERE id = old.id;
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER edit_routes_tg
    INSTEAD OF INSERT OR UPDATE ON manager_routes_verbose
    FOR EACH ROW
    EXECUTE PROCEDURE edit_routes();
```

Проверка соответствия направлений станции и маршрута

```
CREATE OR REPLACE FUNCTION check_route_station() RETURNS TRIGGER AS $$
DECLARE
    d1 directions.name%TYPE;
    d2 directions.name%TYPE;
BEGIN
    SELECT direction INTO d1 FROM stations where id = new.station;
    SELECT direction INTO d2 FROM routes where id = new.route;
    IF d1 <> d2 THEN
        RAISE EXCEPTION 'Станция и маршрут должны относиться к одному направлению: % != %',
d1, d2;
    END IF;
    RETURN new;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER check_route_station_tr
    BEFORE INSERT OR UPDATE ON route_stations
    FOR EACH ROW
    EXECUTE PROCEDURE check_route_station();
```

Проверка изменения направления маршрута, содержащего станции

```
CREATE OR REPLACE FUNCTION check_route_dir_stations() RETURNS TRIGGER AS $$
BEGIN
    IF new.direction <> old.direction AND
        EXISTS (SELECT * FROM route_stations WHERE route = new.id) THEN
        RAISE EXCEPTION 'Не удалось изменить направление: маршрут все еще содержит станции
другого направления.';
    END IF;
    RETURN new;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER check_route_dir_stations_tr
    BEFORE UPDATE ON routes
    FOR EACH ROW
    EXECUTE PROCEDURE check_route_dir_stations();
```

Проверка заведующего направления

```
CREATE OR REPLACE FUNCTION check_manager() RETURNS TRIGGER AS $$
DECLARE
    m RECORD;
BEGIN
    IF new.manager IS NOT NULL THEN
        SELECT post, emp_date, quit_date INTO m FROM employees WHERE tabno = new.manager;
        IF m.post <> 'Заведующий направлением' OR NOT (m.emp_date <= current_date AND
            (m.quit_date IS NULL OR m.quit_date IS NOT NULL AND current_date <=
m.quit_date)) THEN
            RAISE EXCEPTION 'Может быть назначен только активный сотрудник должности
"Заведующий направлением".';
        END IF;
        IF EXISTS (SELECT * FROM directions WHERE manager = new.manager) THEN
            RAISE EXCEPTION 'Сотрудник может заведовать только одним направлением';
        END IF;
    END IF;
    RETURN new;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER check_manager_tr
BEFORE INSERT OR UPDATE ON directions
FOR EACH ROW
EXECUTE PROCEDURE check_manager();
```

Проверка назначения заведующим при смене должности заведующего направлением

```
CREATE OR REPLACE FUNCTION check_manager_emp() RETURNS TRIGGER AS $$
BEGIN
    IF old.post = 'Заведующий направлением' AND new.post <> old.post AND
        EXISTS (SELECT * FROM directions WHERE manager = old.tabno) THEN
        RAISE EXCEPTION 'Не удалось поменять должность: сотрудник все еще является
заведующим направления.';
    END IF;
    RETURN new;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER check_manager_emp_tr
BEFORE UPDATE ON employees
FOR EACH ROW
EXECUTE PROCEDURE check_manager_emp();
```

Вспомогательная функция, рассчитывающая дату и время окончания рейса по дате отправления и маршруту

```
CREATE OR REPLACE FUNCTION ride_end(route_in rides.route%TYPE, ddate_in rides.ddate%TYPE)
RETURNS rides.ddate%TYPE AS $$
DECLARE
    r RECORD;
    way routes.way%TYPE;
    dist stations.distance%TYPE;
    t_end route_stations.arrive_time%TYPE;
BEGIN
    SELECT routes.way INTO way FROM routes WHERE id = route_in;
    SELECT CASE way
        WHEN 'в город' THEN MIN(distance)
        ELSE MAX(distance)
    END
    INTO dist
    FROM route_stations AS rs, stations AS st
    WHERE rs.station = st.id AND rs.route = route_in;
    SELECT arrive_time INTO t_end FROM stations AS st, route_stations AS rs
    WHERE rs.station = st.id AND rs.route = route_in AND distance = dist;
    IF EXTRACT(hour from ddate_in) > 16 AND t_end < '8:00' THEN
        RETURN DATE(ddate_in + '1 day') + t_end;
    ELSE
        RETURN DATE(ddate_in) + t_end;
    END IF;
END;
$$ LANGUAGE plpgsql;
```

Проверка рейса

```
CREATE OR REPLACE FUNCTION check_ride() RETURNS TRIGGER AS $$
DECLARE
    t RECORD;
    m RECORD;
    wdays routes.wdays%TYPE;
BEGIN
    IF new.train IS NOT NULL THEN
        SELECT serv_start_date, serv_end_date INTO STRICT t FROM trains WHERE id =
new.train;
        IF NOT (t.serv_start_date <= current_date AND
            (t.serv_end_date IS NULL OR t.serv_end_date IS NOT NULL AND current_date <=
t.serv_end_date)) THEN
            RAISE EXCEPTION 'Может быть назначен только действующий поезд.';
        END IF;
    END IF;
    IF new.machinist IS NOT NULL THEN
        SELECT emp_date, quit_date, post INTO STRICT m FROM employees WHERE tabno =
new.machinist;
        IF m.post <> 'Машинист' OR NOT (m.emp_date <= current_date AND
            (m.quit_date IS NULL OR m.quit_date IS NOT NULL
            AND current_date <= m.quit_date)) THEN
            RAISE EXCEPTION 'Может быть назначен только действующий сотрудник должности
"Машинист".';
        END IF;
    END IF;
    IF NOT EXISTS (SELECT * FROM route_stations WHERE route = new.route) THEN
        RAISE EXCEPTION 'Не удалось внести изменения: маршрут не содержит станций';
    END IF;
    SELECT routes.wdays INTO STRICT wdays FROM routes WHERE id = new.route;
```



```

IF wdays = 'по рабочим' THEN
    IF EXTRACT(dow FROM new.ddate) NOT IN (1, 2, 3, 4, 5) THEN
        RAISE EXCEPTION 'День недели не соответствует режиму движения: %', wdays;
    END IF;
ELSIF wdays = 'по выходным' THEN
    IF EXTRACT(dow FROM new.ddate) NOT IN (6, 0) THEN
        RAISE EXCEPTION 'День недели не соответствует режиму движения: %', wdays;
    END IF;
END IF;
IF EXISTS (SELECT * FROM rides WHERE id <> new.id AND
            (machinist = new.machinist OR train = new.train) AND
            (new.ddate BETWEEN ddate AND ride_end(route, ddate) OR
             ddate BETWEEN new.ddate AND ride_end(new.route, new.ddate))) THEN
    RAISE EXCEPTION 'Не удалось внести изменения: рейсы машиниста или поезда
пересекаются.';
END IF;
RETURN new;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        IF t IS NULL THEN
            RAISE EXCEPTION 'Не удалось найти поезд %', new.train;
        ELSIF m IS NULL THEN
            RAISE EXCEPTION 'Не удалось найти сотрудника %', new.machinist;
        ELSIF wdays IS NULL THEN
            RAISE EXCEPTION 'Не удалось найти маршрут %', new.route;
        END IF;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER check_ride_tr
    BEFORE INSERT OR UPDATE ON rides
    FOR EACH ROW
    EXECUTE PROCEDURE check_ride();

```

Проверка назначения на рейс при смене должности машиниста

```

CREATE OR REPLACE FUNCTION check_machinist_emp() RETURNS TRIGGER AS $$
BEGIN
    IF old.post = 'Машинист' AND new.post <> old.post AND
        EXISTS (SELECT * FROM rides WHERE machinist = old.tabno AND ddate >=
current_timestamp) THEN
        RAISE EXCEPTION 'Не удалось поменять должность: машинист назначен на будущий
рейс.';
    END IF;
    RETURN new;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER check_machinist_emp_tr
    BEFORE UPDATE ON employees
    FOR EACH ROW
    EXECUTE PROCEDURE check_machinist_emp();

```

Проверка назначения на рейс при редактировании поезда

```
CREATE OR REPLACE FUNCTION check_train_upd() RETURNS TRIGGER AS $$
BEGIN
    IF (old.serv_end_date IS NULL AND new.serv_end_date IS NOT NULL OR
        old.serv_end_date IS NOT NULL AND old.serv_end_date <> new.serv_end_date) AND
        EXISTS (SELECT * FROM rides WHERE train = new.id AND ddate >=
new.serv_end_date) THEN
        RAISE EXCEPTION 'Не удалось изменить дату: поезд все еще назначен на будущий рейс';
    END IF;
    RETURN new;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER check_train_upd_tr
BEFORE UPDATE ON trains
FOR EACH ROW
EXECUTE PROCEDURE check_train_upd();
```

2.6 Создание индексов

Индекс по ВНК таблицы “Направления” к таблице “Сотрудники” целесообразен при увеличении числа сотрудников для ускорения представлений заведующего:

```
CREATE INDEX fk_directions_manager ON directions(manager);
```

Индекс по ВНК таблицы “Сотрудники” к таблице “Должности” целесообразен при увеличении числа должностей, так как выборка по должности производится в функциях и представления довольно часто:

```
CREATE INDEX fk_employees_post ON employees(post);
```

Для ускорения соединения часто участвующих в соединении таблиц могут быть полезны индексы и по другим внешним ключам.

```
CREATE INDEX fk_stations_direction ON stations(direction);
CREATE INDEX fk_rides_route ON rides(route);
CREATE INDEX fk_rides_train ON rides(train);
CREATE INDEX fk_rides_machinist ON rides(machinist);
```

Таблица “Билеты” обновляется очень часто, к тому же действие добавления производится с ней много чаще, чем выборки, поэтому индексы в ней нецелесообразны.

Таблица “Рейсы” также обновляется чаще других, но индекс по дате в ней может быть очень полезен (для получения расписания):

```
CREATE INDEX ind_rides_ddate ON rides(ddate);
```

Для ускорения представлений типа “Активный штат” и “Активные поезда” полезны индексы по датам:

```
CREATE INDEX ind_employees_period ON employees(emp_date, quit_date);
CREATE INDEX ind_trains_period ON trains(serv_start_date, serv_end_date);
```

При поиске сотрудников может быть полезен индекс по ФИО:

```
CREATE INDEX ind_employees_fio ON employees(last_name, first_name, patronymic);
```

2.7 Разработка стратегии резервного копирования

Интенсивность обновления разработанной базы данных довольно высока, наиболее часто обновляемыми являются таблицы “Билеты” и “Рейсы” (в меньшей степени), поэтому их изменения рекомендуется сохранять как можно чаще. Для обеспечения сохранности данных рекомендуется производить полное резервное копирование БД хотя бы раз в день, в ночное время, когда движение поездов и работа кассиров приостанавливается.

3. Интерфейс

3.1 Используемые технологии

При создании клиентского приложения использованы следующие программные решения:

- python 3.7 в качестве основного языка программирования
- tkinter – библиотека для создания GUI с помощью python
- sqlalchemy – библиотека для работы с реляционными БД

3.2 Описание

Основное окно клиентского приложения содержит две области: область взаимодействия (кнопки) и область просмотра (таблицы).

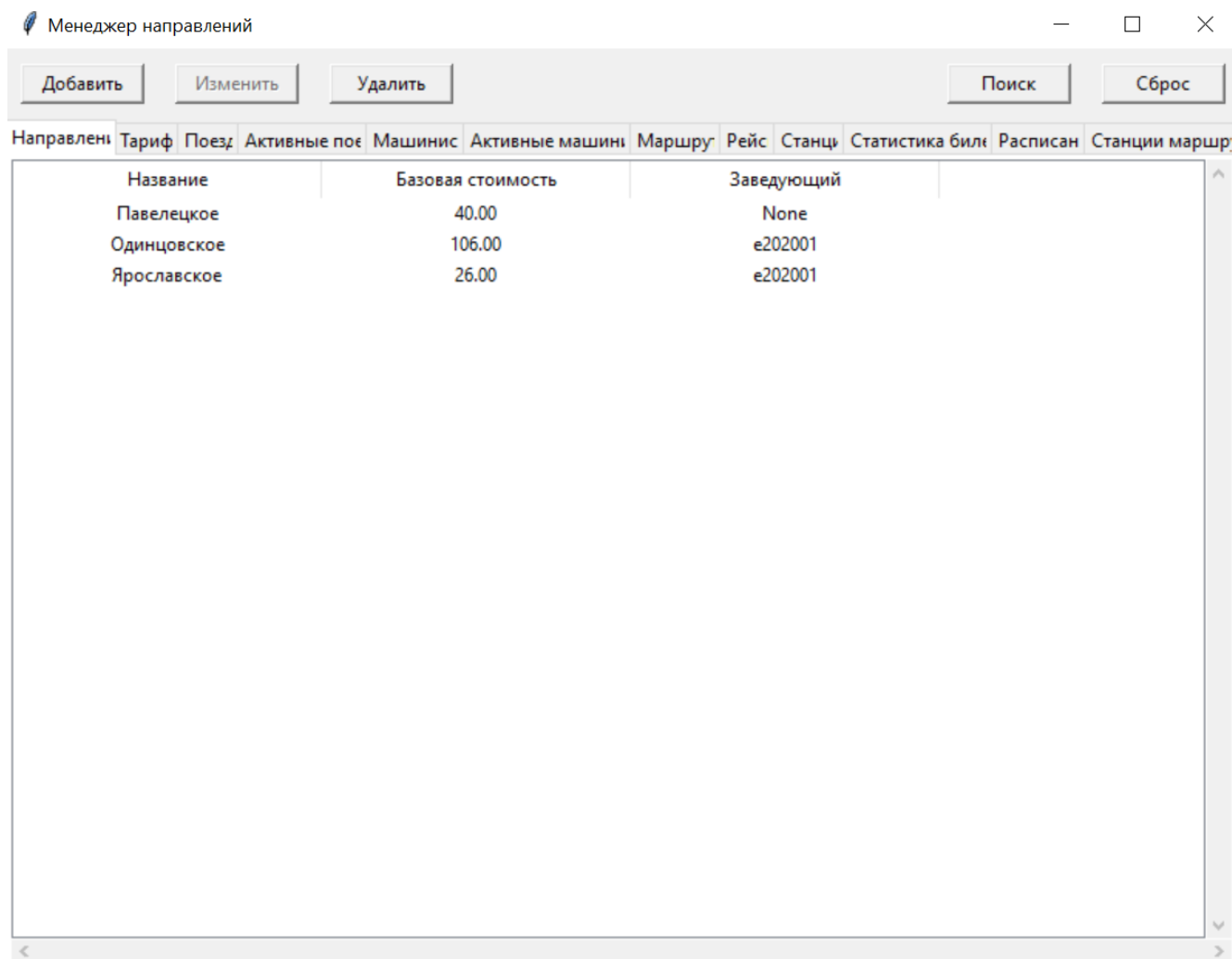


Рис. 4 Общий вид приложения

Перед открытием основного окна пользователю предлагается авторизоваться по табельному номеру сотрудника.

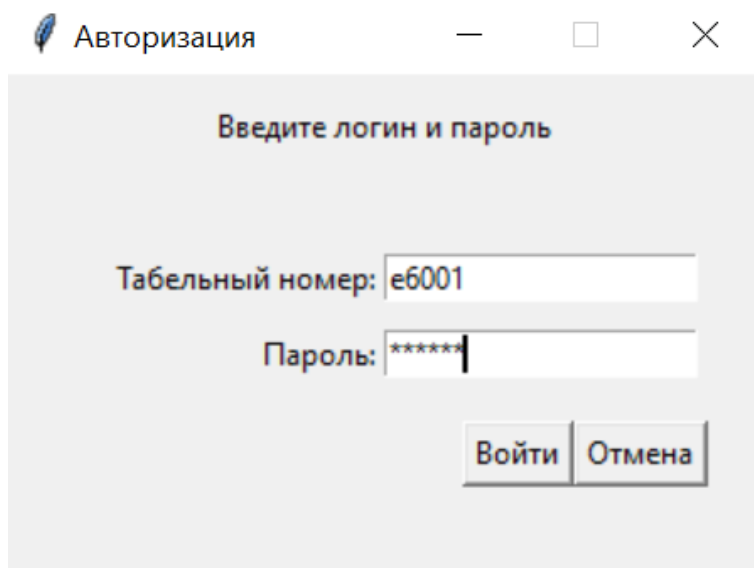
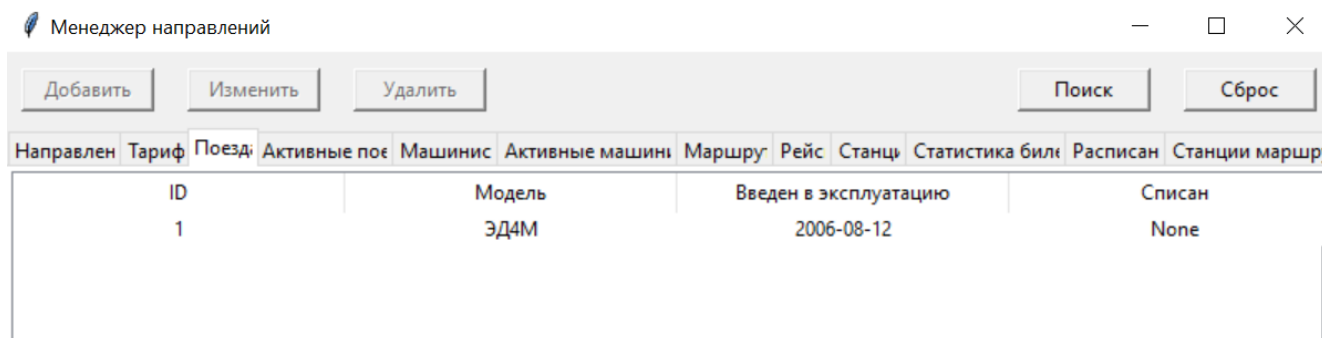


Рис. 5 Окно авторизации

В зависимости от должности авторизованного сотрудника отображаются таблицы и представления в соответствии с доступными ему действиями над БД. В зависимости от прав доступа должности сотрудника меняется набор доступных кнопок области взаимодействия.



ID	Модель	Введен в эксплуатацию	Списан
1	ЭД4М	2006-08-12	None

Рис. 6 Пример главного окна роли “Менеджер направлений”

3.3 Демонстрация

Главные окна для сотрудников должностей “Кассир” и “Машинист”

Кассир

Добавить Изменить Удалить Поиск Сброс

Расписание Билеты

Дата	Номер поезда	Номер маршрута	Тариф
2021-05-17 08:30:00	1	398	стандарт

Рис. 7 Пример главного окна роли “Кассир”

Машинист

Добавить Изменить Удалить Поиск Сброс

Назначенные рейсы Расписание

Дата	Поезд	Номер маршрута	Тип
2021-05-17 08:30:00	1	398	стандарт

Рис. 8 Пример главного окна роли “Машинист”

Добавление записи

Менеджер направлений

Добавить Изменить Удалить Поиск Сброс

Направлен Тариф Поезд Активные поезда Машинист Активные машины Маршрут Рейс Станция Статистика билета Расписание Станции маршрута

ID	Направление	Тип	Режим движения
398	Ярославское	стандарт	по выходным

Введите значения

ID: 399

Направление: Ярославское

Тип: стандарт

Режим движения: ежедневно

Сторона:
в город
из города

OK

Рис. 9 Пример окна добавления

Менеджер направлений

Добавить Изменить Удалить Поиск Сброс

Направлен Тариф Поезд Активные поезда Машинист Активные машины Маршрут Рейс Станция Статистика билета Расписание Станции маршрута

ID	Направление	Тип	Режим движения
398	Ярославское	стандарт	по выходным
399	Ярославское	стандарт	ежедневно

Рис. 10 Результат добавления

Добавление ошибочной записи

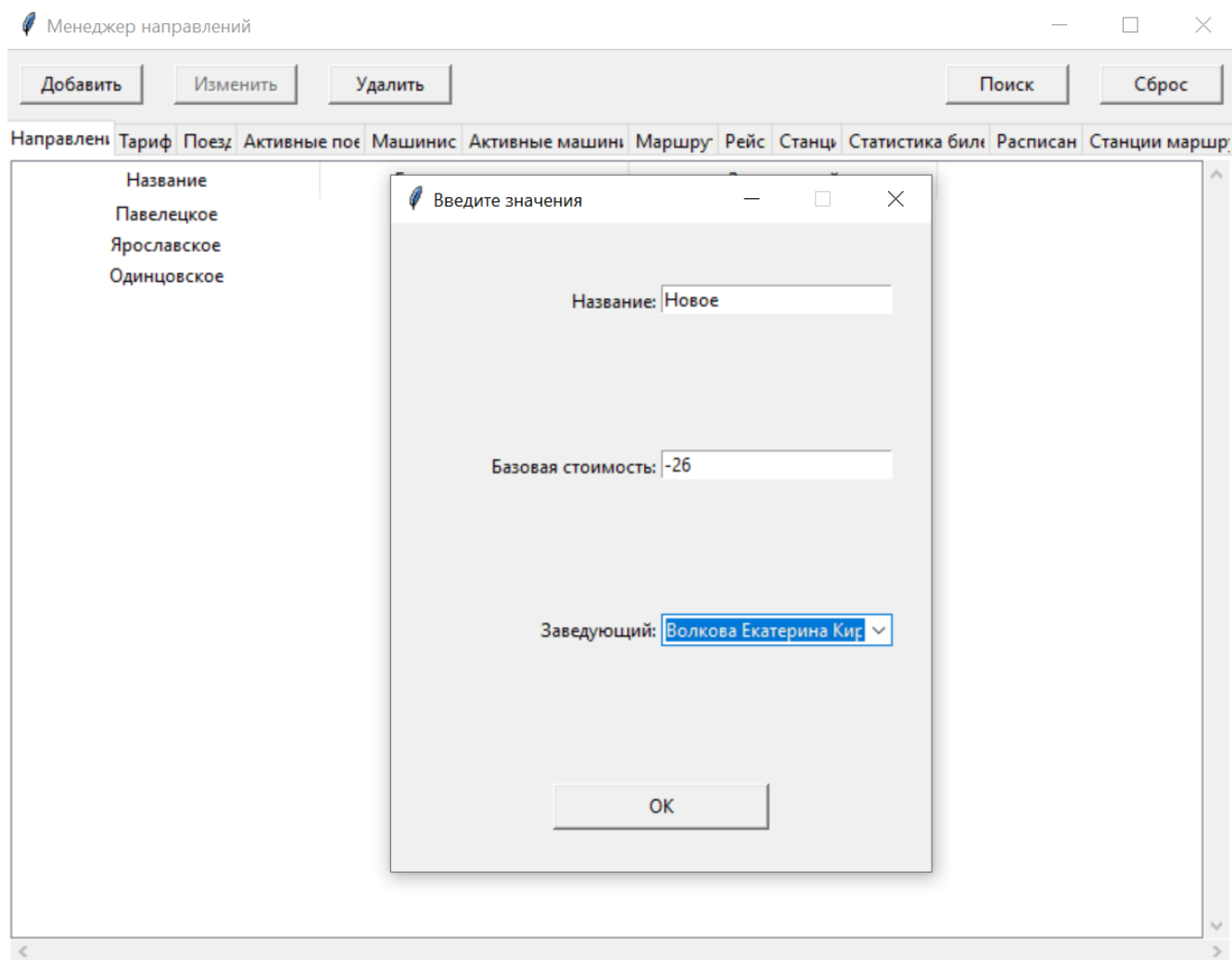


Рис. 11 Пример добавления недопустимой записи

При нарушении ограничений целостности БД отображается окно с описанием ошибки:

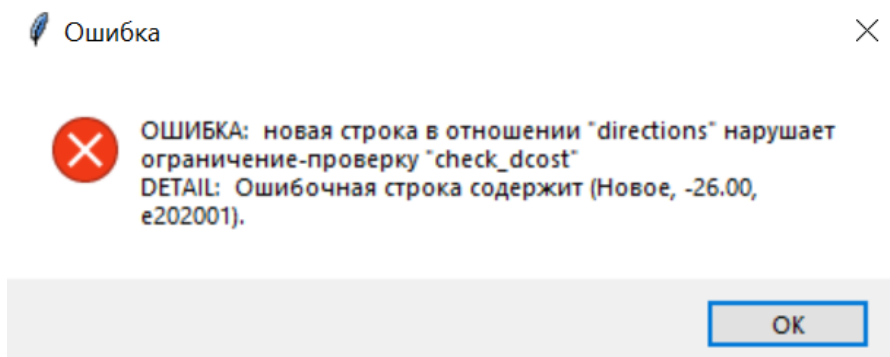


Рис. 12 Пример ошибки

Редактирование записи

Менеджер направлений

Добавить Изменить Удалить Поиск Сброс

Направлен Тариф Поезд Активные поезда Машинист Активные машины Маршрут Рейс Станция Статистика билета Расписание Станции маршрута

ID	Направление	Тип	Режим движения по выходным
398			
399			ежедневно

Введите значения

ID: 399

Направление: Ярославское

Тип: экспресс

Режим движения: ежедневно

Сторона: в город

OK

Рис. 13 Пример окна редактирования записи

Менеджер направлений

Добавить Изменить Удалить Поиск Сброс

Направлен Тариф Поезд Активные поезда Машинист Активные машины Маршрут Рейс Станция Статистика билета Расписание Станции маршрута

ID	Направление	Тип	Режим движения
398	Ярославское	стандарт	по выходным
399	Ярославское	экспресс	ежедневно

Рис. 14 Результат редактирования

Поиск по записям

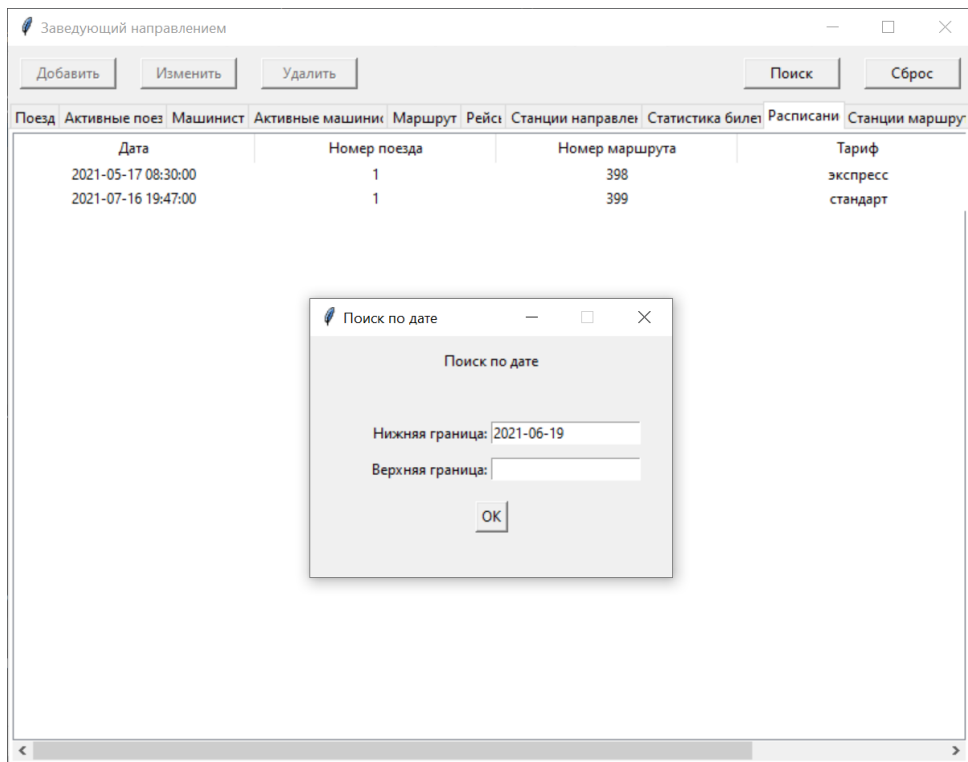


Рис. 15 Пример окна поиска по дате

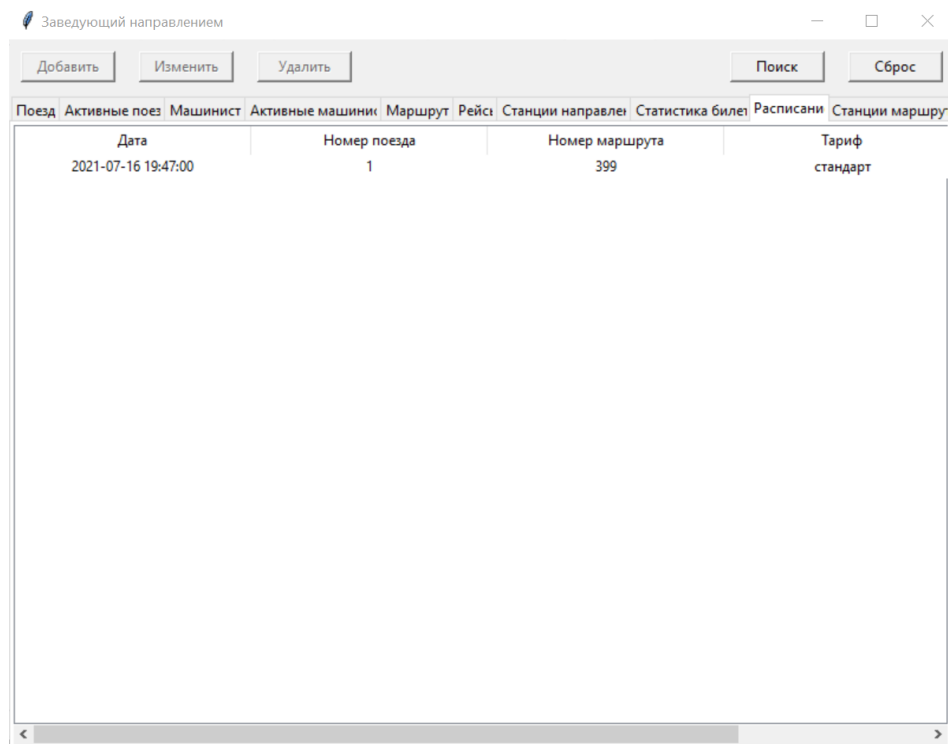


Рис. 16 Результат поиска по дате

Кнопка “Сброс” сбрасывает параметры поиска и загружает данные текущей таблицы из БД.

3.4 Реализация

Полный код приложения содержится в открытом репозитории:

https://github.com/KonstantAnxiety/suburban_trains_db