ВВЕДЕНИЕ

Метрополитен всегда нуждался и будет нуждаться в автоматизации. От того насколько эффективно и точно составляется расписание, выявляются неполадки в подвижных составах, зависят жизни людей и благополучная транспортная обстановка в городе.

Целью данной работы является разработка информационной системы с клиент-серверной архитектурой, а так же клиентского приложения. Разрабатываемая ИС должна хранить информацию о метрополитене. Ценность системы автоматизации службы движения метрополитена очевидна: отпадает необходимость вручную высчитывать оптимальные интервалы движения подвижных составов и составлять расписание на основе этих данных, что уменьшает риски возникновения чрезвычайных ситуаций.

1 Анализ задачи

1.1 Семантическое описание предметной области

Предметная область: Служба движение метрополитена

Данная информационная система разрабатывается для обеспечения автоматизации системы метрополитена. Целью данного режимного объекта, как любой организации, является минимизация затрат и обеспечение безопасности.

Задачей проекта является организация для метрополитена наиболее эффективного способа ведения информации о сотрудниках, движении составов.

В базе данных требуется хранить информацию о следующих объектах:

1) Подвижных составах;
2) Типах подвижных составов;
3) Станциях метрополитена;
4) Ветках метрополитена;
5) Сотрудниках: Машинистах, Диспетчеров;
6) Ремонтном депо;
7) Справочной таблице поломок;

Необходимо учитывать следующие бизнес правила;

1) Серийный номер состава должен быть уникальным;
2) Минимальный интервал между составами должен быть 90 секунд;
3) Машинист может управлять только одним подвижным составом;

4) Один подвижной состав может управляться несколькими машинистами;

5) Номер паспорта у сотрудника должен быть уникальным;

6) Код станции должен быть уникальным;

7) Код типа состава должен быть уникальным;

8) Номер тупика у ремонтной станции должен быть уникальным;

9) Код поломки в справочной таблице должен быть уникальным;

10) Первая станция в ветке является депо, нельзя добавить станцию перед первой станцией;

11) По ветке могут ходить подвижные составы только одного типа;

12) Статусы для депо ветке и движения по ветке должны быть уникальными;

1.2 Выявление потребностей пользователя ИС

Функциональную модель информационной системы можно наглядно представить в виде диаграммы вариантов использования (рисунок 1).

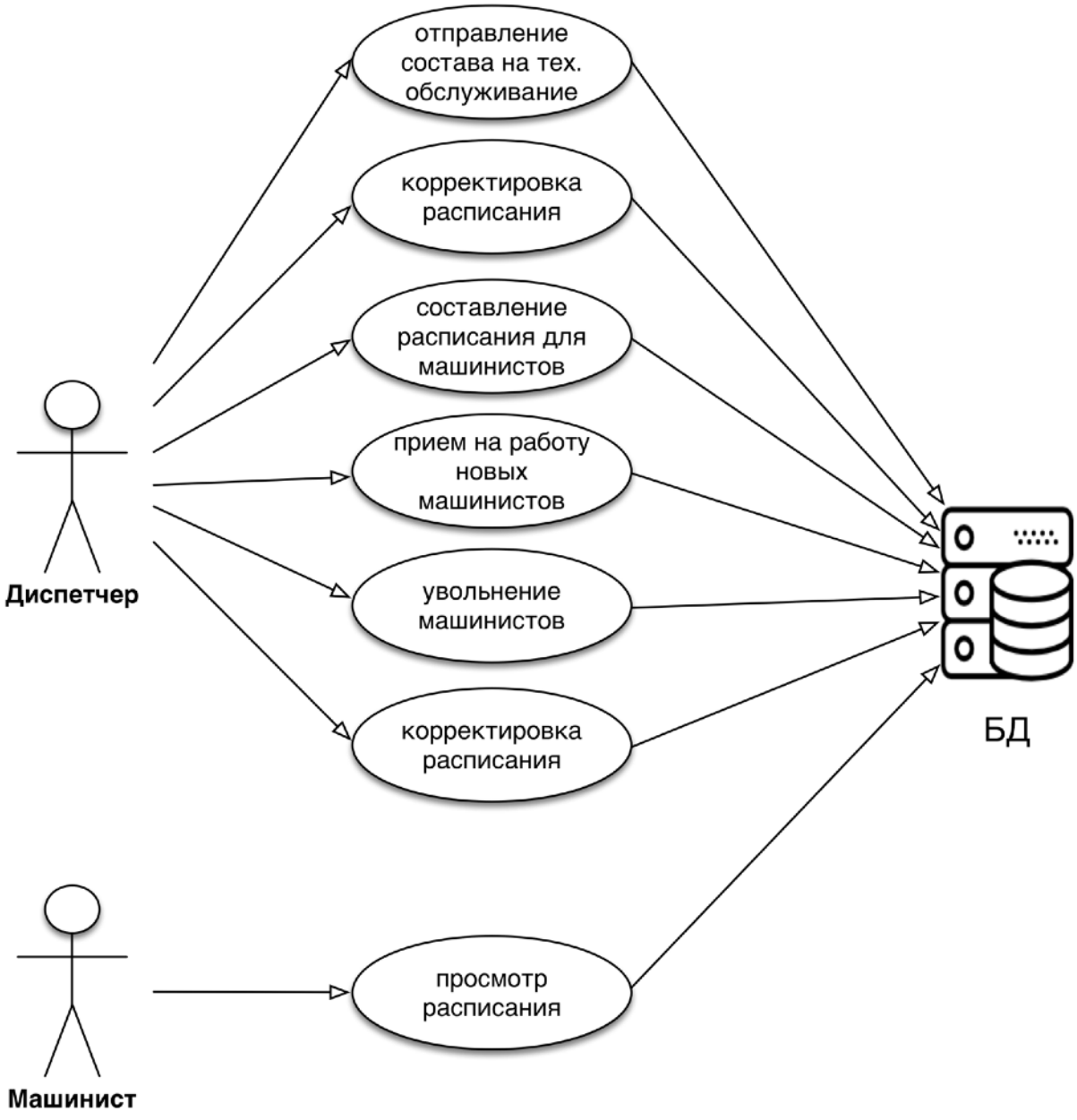


Рисунок 1. Диаграмма вариантов использования

1.3 Анализ существующих решений

В России существует 7 действующих метрополитена: Московский, Петербургский, Нижегородский, Казанский, Новосибирский, Самарский, Екатеринбургский.

Все они является режимными объектами, поэтому информация о том какие программные комплексы используются не доступна без необходимого уровня допуска.

1.4 Разработка общей структуры ИС

В разрабатываемой информационной системе будет использована архитектура типа «Клинт-Сервер». Клиентами являются ПК диспетчеров. Все запросы, которые будут осуществлены через клиентское приложение, передаются клиентом на сервер БД, где и происходит обработка данных. Предполагаемая модель выбрана для уменьшения нагрузки на клиентские ПК и облегчит процесс поиска ошибок, если такие обнаружатся.

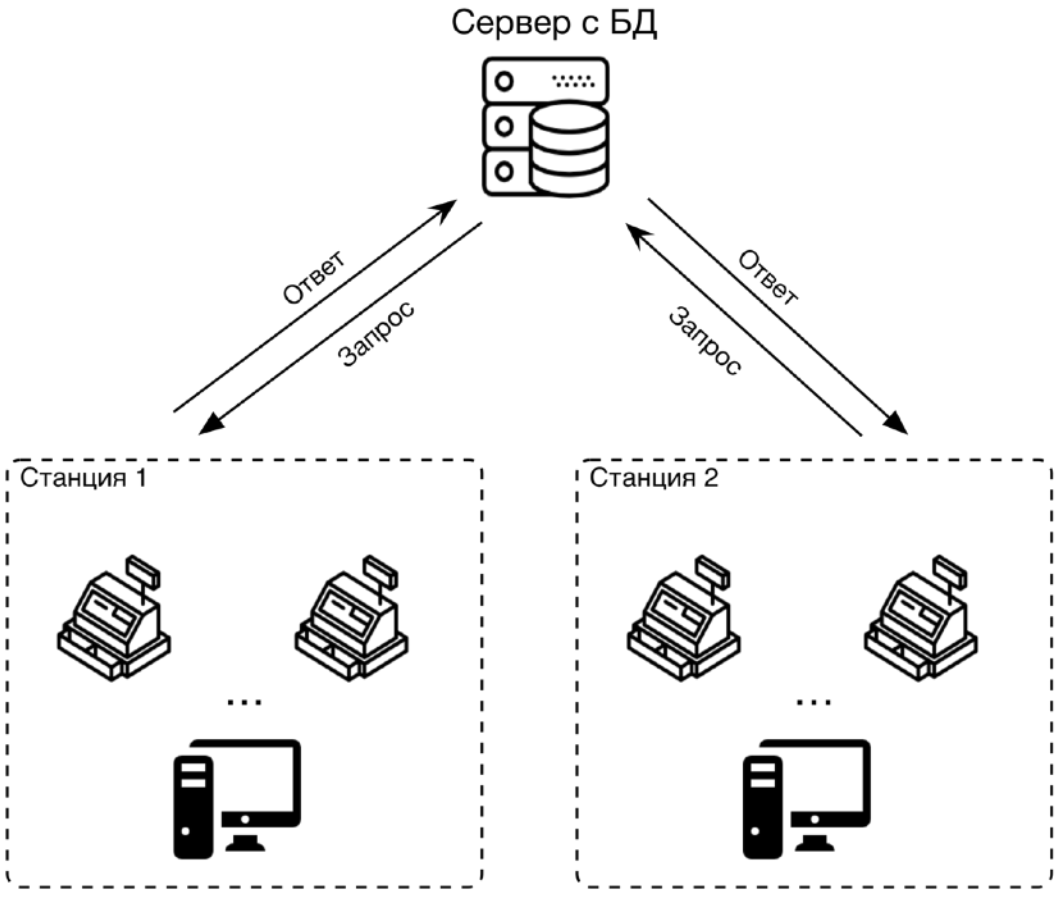


Рисунок 2. Архитектура «Клиент-Сервер»

2 Разработка серверной части ИС

2.1 Инфологическое проектирование БД

2.1.1 Выявление сущностей и связей

В данной предметной области можно выделить следующие сущности:

1) Машинист (Номер_Паспорта);
2) Диспетчер (Номер_Паспорта);
3) Подвижной состав (Код_Состава);
4) Тип Состава (Код_Типа);
5) Ремонтная станция (Номер_Тупика);
6) Станция (Код_Станции);
7) Ветка (Название_Ветки);
8) Справочная поломок (Код_Поломки);

В предметной области можно выделить следующие отношения:

1) Диспетчер добавляет Подвижные составы в расписание для Станций;
2) Подвижной состав имеет Тип Состава;
3) Диспетчер отправляет Подвижной состав на Ремонтную станцию;
4) Машинист управляет Подвижным составом;
5) Ветка состоит из Станций;

2.1.2 Построение ER-диаграмм классов

На основе отношений, описанных выше, строятся ER-диаграммы классов.

1) Диспетчер добавляет Подвижные составы в расписание для Станций.

Для степени связи:

- Много Диспетчеров добавляют много Подвижных Составов в расписание для многих Станций.

Для класса принадлежности сущности:

- Диспетчер не обязательно добавляет Подвижной Состав в расписание.
- Подвижной состав не обязательно добавляется для Станции.
- Станция не обязательно имеет Подвижной Состав.

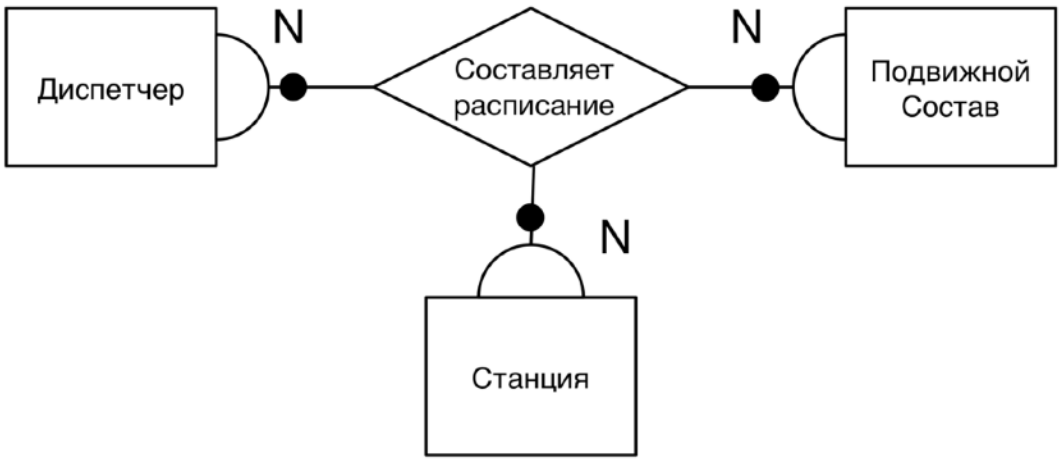


Рисунок 3. ER-сегмент Диспетчер - Подвижной состав - Станция

2) Подвижной состав имеет Тип Состава.

Для степени связи:

- Подвижной состав может иметь только один Тип Состава.
- Тип Состава имеет много Подвижных Составов.

Для класса принадлежности сущность к связи:

- Подвижной состав обязательно имеет тип.
- Тип Состава не обязательно имеет Подвижной Состав.

Рисунок 4. ER-сегмент Подвижной Состав - Тип Состава

3) Диспетчер отправляет Подвижной состав в Ремонтную Станцию.

Для степени связи:

- Много Диспетчеров отправляют много Подвижных составов на множество Ремонтов.

Для класса принадлежности сущность к связи:

- Подвижной Состав не обязательно отправляется на Ремонтную станцию.
- На ремонтной станции не обязательно находится Подвижной Состав.
- Диспетчер не обязательно отправляет Подвижной Состав.

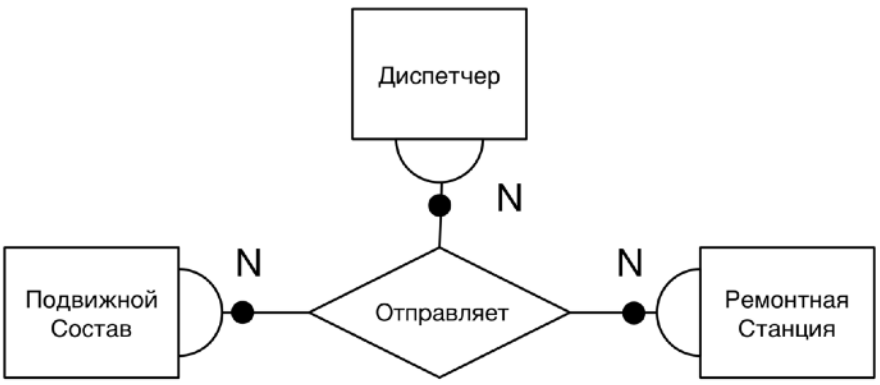


Рисунок 5. ER-сегмент Подвижной Состав - Диспетчер - Ремонтная Станция

4) Машинист управляет подвижным составом.

Для степени связи:

- Машинист управляет только одним Подвижным Составом.
- Подвижным составом могут управлять несколько Машинистов.

Для класса принадлежности сущность к связи:

- Машинист обязательно управляет Подвижным Составом.
- Подвижной состав не обязательно управляется Машинистом.

Рисунок 6. ER-сегмент Машинист - Подвижной Состав

5) Ветка состоит из Станций.

Для степени связи:

- Многие станции принадлежат одной ветке.

Для класса принадлежности сущности к связи:

- Ветке обязательно принадлежит хотя бы одна станция.
- Станция не обязательно принадлежит ветке.

Рисунок 7. ER-сегмент Ветка - Станция

5) Ремонтная станция имеет Справочную поломок.

Для степени связи:

- Одна Ремонтная станция имеет множество поломок.

Для класса принадлежности сущности к связи:

- Ремонтная станция обязательно имеет Справочную поломок.
- Справочная поломок не обязательно имеет Ремонтную станцию.

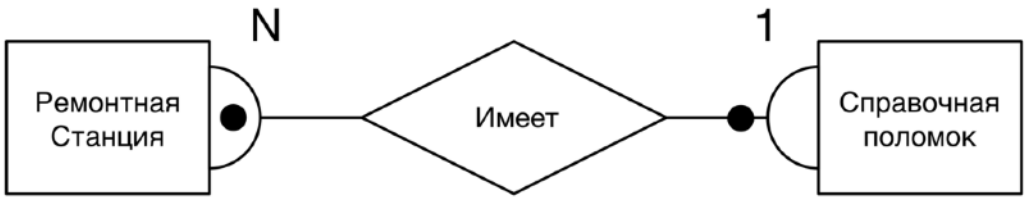


Рисунок 8. ER-сегмент Ремонтная Станция - Справочная поломок

На основе сегментов ER-диаграммы, построена общая ER-диаграмма. Она представлена на рисунке 9.

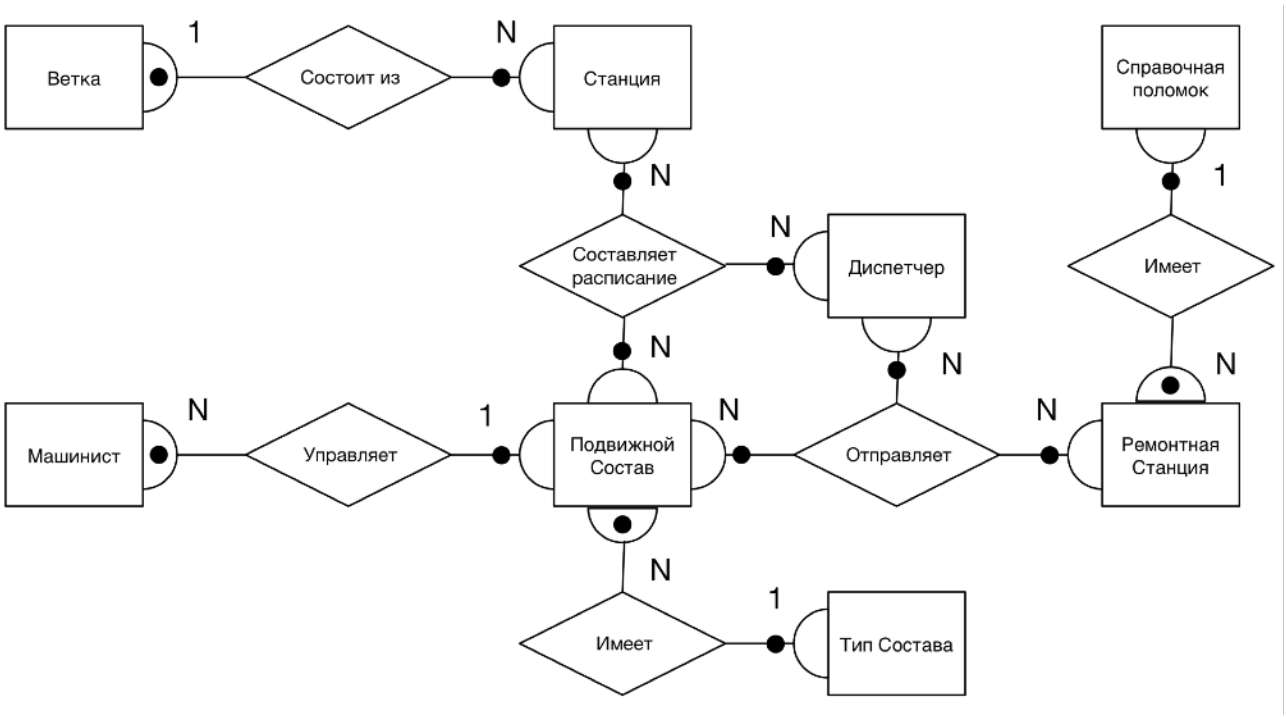


Рисунок 9. Общая ER - диаграмма

2.2 Даталогическое проектирование БД

Формирование набора предварительных отношений по ER-диаграмме с указанием предполагаемого первичного ключа для каждого отношения.

Для перехода от ER-диаграммы к предварительным отношениям воспользуемся правилами формирования отношений для бинарных связей «один-ко-многим», «многие-ко-многим» и тернарных связей «многие-ко-многим».

Далее приведены формулировки правил:

1) Если степень бинарной связи 1:N и класс принадлежности N-связной сущности обязательный, то формируется два отношения, по одному для каждой сущности. Ключ отношения - это ключ соответствующей сущности. Кроме этого в отношение для N-связной сущности добавляется в качестве атрибута ключ односвязной сущности.

2) Если степень бинарной связи N:M, то формируется три отношения: два отношения для сущностей, где ключом является ключ соответствующей сущности и одно отношение для связи, где первичный ключ состоит из ключей обеих сущностей.

3) При наличии тернарной связи необходимо использовать 4 отношения: три для каждой сущности, где ключом является ключ соответствующей сущности и одно для связи, где ключ содержит ключи всех сущностей. При наличии n-ой связи строится n+1 отношение.

2.2.1 Переход от ER-диаграмм к предварительным отношениям

По правилам 1, 2 и 3 были выделены следующие предварительные отношения:

1) Диспетчер добавляет Подвижные составы в расписание для Станций.

По правилу 3 получим:

- Диспетчер (Номер_Паспорта)
- Подвижной состав (Серийный_Номер)
- Станция (Код_Станции)
- Расписание (Дата_Время, Код_Станции, Серийный_Номер, Номер_Паспорта)

2) Подвижной Состав имеет Тип Состава.

По правилу 1 получаем:

- Подвижной состав (Серийный_Номер, Код_Типа)
- Тип Состава (Код_Типа)

3) Диспетчер отправляет Подвижной состав на Ремонтную станцию.

По правилу 3 получаем:

- Диспетчер (Номер_Паспорта)
- Подвижной Состав (Серийный_Номер)
- Ремонтная станция (Номер_Тупика)
- Журнал ремонта (НачалоРемонта, Номер_Паспорта, Серийный_Номер, Номер_Тупика)

4) Машинист управляет Подвижным Составом.

По правилу 1 получаем:

- Машинист (Номер_Паспорта, Серийный_Номер)
- Подвижной Состав (Серийный_Номер)

5) Ветка состоит из Станций.

По правилу 1 получаем:

- Ветка (Название, Код_станции)
- Станция (Код_станции)

2.2.2 Заполнение предварительных отношений атрибутами

Заполним предварительные отношения атрибутами. В скобках перечислены атрибуты соответствующих отношений:

- Машинист (Номер_Паспорта, Серийный_Номер, ФИО, Телефон, Стаж, Квалификация)
- Диспетчер (Номер_Паспорта, ФИО, Телефон, Должность, Пароль)
- Станция (Код_Станции, Название_Станции, Пассажиропоток, Расстояние_вперед, Расстояние_назад, Код_Станции_впереди, Код_Станции_позади, Статус_Станции, Время_открытия, Время_закрытия)
- Ветка (Название, Начальная_Станция, Конечная_Станция, Статус_депо, Статус_движения_вперед, Статус_движения_назад)
- Подвижной Состав (Серийный_Номер, Код_Типа, Статус, Пробег, Изношеность_мотора, Изношенность_ТК)
- Тип состава (Код_Типа, Название, Количество_Мест, Длина, Ширина, Высота, Колея, Назначение, Скорость)
- Ремонтная_Станция (Номер_Тупика, Серийный_Номер, Дата_Начала, Код_Поломки)
- Справочная поломок (Код_Поломки, Название_поломки, Критическое_Значение, Срок_Доставки_Детали, Примерное_Время_Ремонта)
- Расписание (Время_отправления, Код_Станции_отправления, Серийный_Номер_состава, Номер_Паспорта, Время_прибытия, Станция_прибытия, Ветка, Направление)

- Журнал ремонта (Номер_Паспорта, Серийный_Номер, Номер_Тупика, Дата_Время_Начала, Заметка)

2.2.3 Проверка предварительных отношений на соответствие нормальным формам

Необходимо проверить отношения на соответствие нормальным формам, в связи с тем, что при несоответствии отношений нормальным формам могут возникать аномалии модификации и избыточность данных:

1) Машинист (Номер_Паспорта, Серийный_Номер, ФИО, Телефон, Стаж, Квалификация)

- отношение находится в 2НФ, так как ключ простой и, соответственно, каждый не ключевой атрибут функционально полно зависит от потенциального ключа;
- отношение находится в 3НФ, так как отсутствуют транзитивные зависимости;
- отношение находится в БКНФ, так как детерминанты всех функциональных зависимостей являются потенциальным ключом.

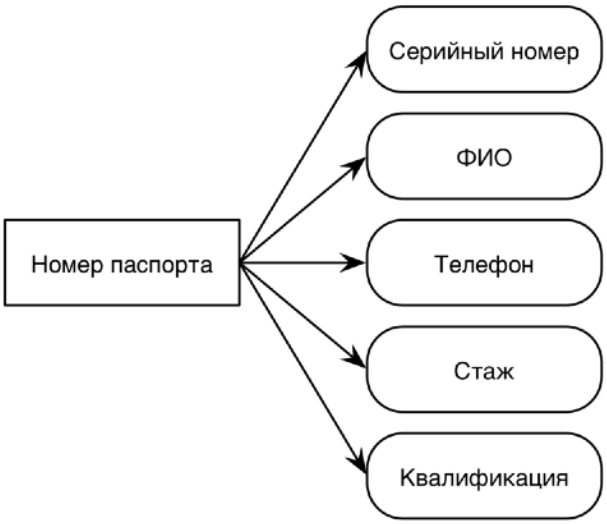


Рисунок 10. Отношение «Машинист»

2) Диспетчер (Номер Паспорта, ФИО, Телефон, Должность, Пароль)

- отношение находится в 2НФ, так как ключ простой и, соответственно, каждый не ключевой атрибут функционально полно зависит от потенциального ключа;
- отношение находится в 3НФ, так как отсутствуют транзитивные зависимости;
- отношение находится в БКНФ, так как детерминанты всех функциональных зависимостей являются потенциальным ключом.

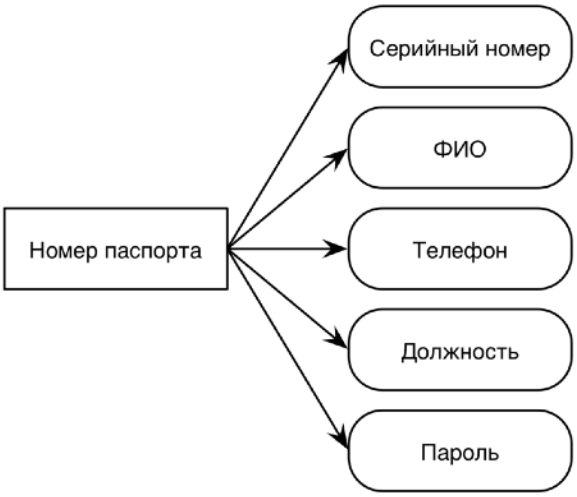


Рисунок 11. Отношение «Диспетчер»

3) Станция (Код_Станции, Название_Станции, Пассажиропоток, Расстояние_вперед, Расстояние_назад, Код_Станции_впереди, Код_Станции_позади, Статус_Станции, Время_открытия, Время_закрытия)

- отношение находится в 2НФ, так как ключ простой и, соответственно, каждый не ключевой атрибут функционально полно зависит от потенциального ключа;
- отношение находится в 3НФ, так как отсутствуют транзитивные зависимости;

- отношение находится в БКНФ, так как детерминанты всех функциональных зависимостей являются потенциальным ключом.

Рисунок 12. Отношение «Станция»

4) Подвижной Состав (Серийный_Номер, Код_Типа, Статус, Пробег, Изношеность_мотора, Изношенность_ТК)

- отношение находится в 2НФ, так как ключ простой и, соответственно, каждый не ключевой атрибут функционально полно зависит от потенциального ключа;
- отношение находится в 3НФ, так как отсутствуют транзитивные зависимости;
- отношение находится в БКНФ, так как детерминанты всех функциональных зависимостей являются потенциальным ключом.

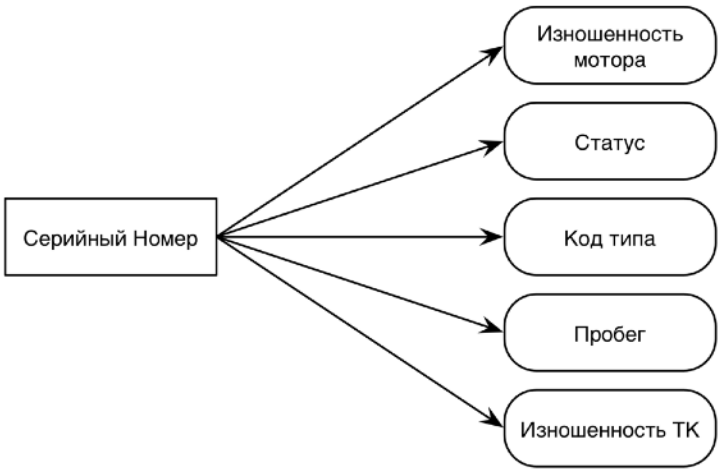


Рисунок 13. Отношение «Подвижной Состав»

5) Тип состава (Код_Типа, Название, Количество_Мест, Длина, Ширина, Высота, Колея, Назначение, Скорость)

- отношение находится в 2НФ, так как ключ простой и, соответственно, каждый не ключевой атрибут функционально полно зависит от потенциального ключа;
- отношение находится в 3НФ, так как отсутствуют транзитивные зависимости;
- отношение находится в БКНФ, так как детерминанты всех функциональных зависимостей являются потенциальным ключом.

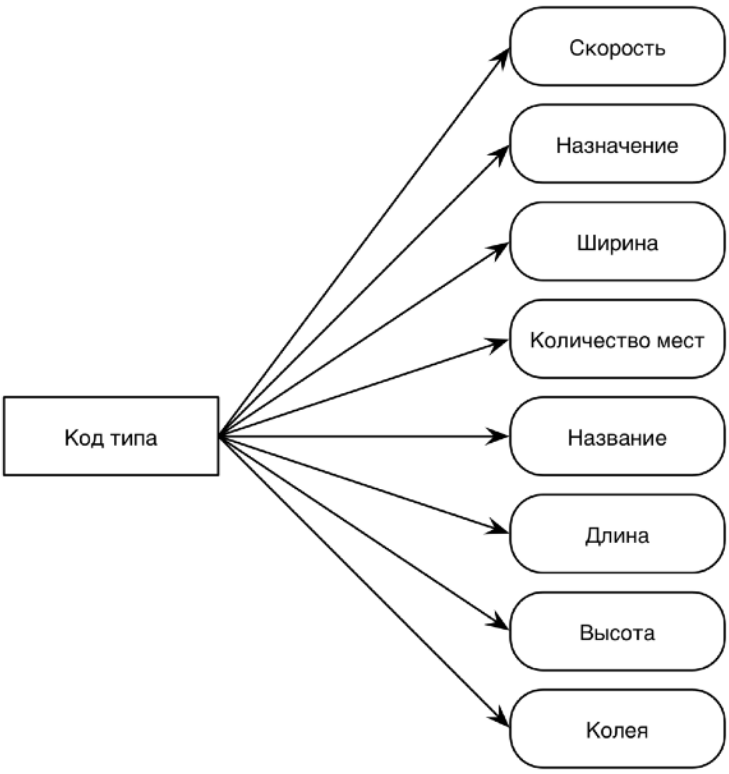


Рисунок 14. Отношение «Тип состава»

6) Справочная поломок (Код Поломки, Название_поломки, Критическое_Значение, Срок_Доставки_Детали, Примерное_Время_Ремонта)

- отношение находится в 2НФ, так как ключ простой и, соответственно, каждый не ключевой атрибут функционально полно зависит от потенциального ключа;
- отношение находится в 3НФ, так как отсутствуют транзитивные зависимости;
- отношение находится в БКНФ, так как детерминанты всех функциональных зависимостей являются потенциальным ключом.

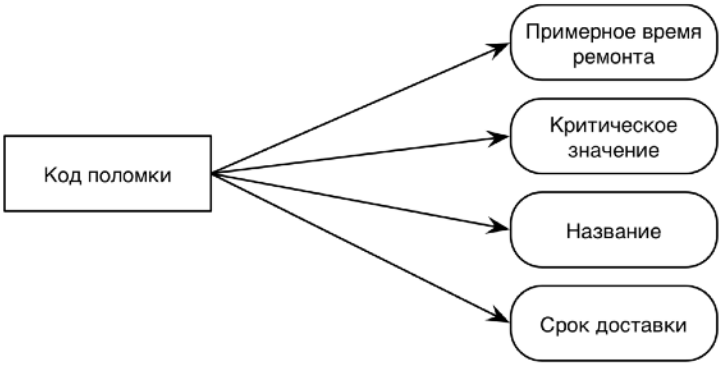


Рисунок 15. Отношение «Справочная поломок»

7) Расписание (Время отправления, Код Станции отправления, Серийный_Номер_состава, Номер Паспорта, Время_прибытия, Станция_прибытия, Ветка, Направление)

- отношение находится в 2НФ, так как ключ простой и, соответственно, каждый не ключевой атрибут функционально полно зависит от потенциального ключа;
- отношение находится в 3НФ, так как отсутствуют транзитивные зависимости;
- отношение находится в БКНФ, так как детерминанты всех функциональных зависимостей являются потенциальным ключом.

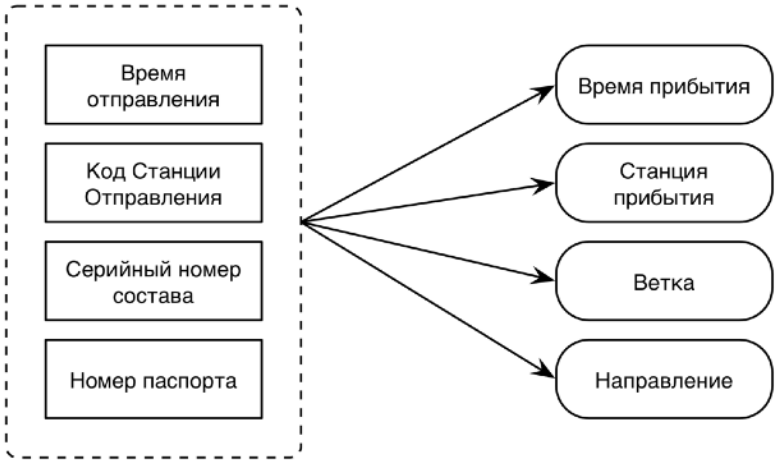


Рисунок 16. Отношение «Расписание»

8) Журнал ремонта (Номер_Паспорта, Серийный_Номер, Номер_Тупика, Дата_Время_Начала, Заметка)

- отношение находится в 2НФ, так как ключ простой и, соответственно, каждый не ключевой атрибут функционально полно зависит от потенциального ключа;
- отношение находится в 3НФ, так как отсутствуют транзитивные зависимости;
- отношение находится в БКНФ, так как детерминанты всех функциональных зависимостей являются потенциальным ключом.

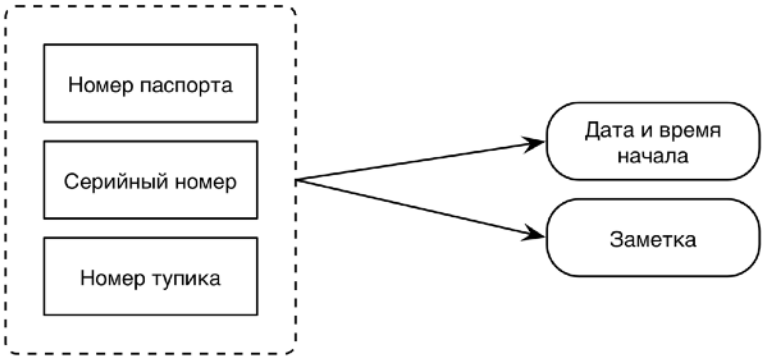


Рисунок 17. Отношение «Журнал ремонта»

9) Ветка (Название, Начальная_Станция, Конечная_Станция, Статус_депо, Статус_движения_вперед, Статус_движения_назад)

- отношение находится в 2НФ, так как ключ простой и, соответственно, каждый не ключевой атрибут функционально полно зависит от потенциального ключа;
- отношение находится в 3НФ, так как отсутствуют транзитивные зависимости;
- отношение находится в БКНФ, так как детерминанты всех функциональных зависимостей являются потенциальным ключом.

Рисунок 18. Отношение «Ветка»

10) Ремонтная_Станция (Номер_Тупика, Серийный_Номер, Дата_Начала, Код_Поломки)

- отношение находится в 2НФ, так как ключ простой и, соответственно, каждый не ключевой атрибут функционально полно зависит от потенциального ключа;
- отношение находится в 3НФ, так как отсутствуют транзитивные зависимости;
- отношение находится в БКНФ, так как детерминанты всех функциональных зависимостей являются потенциальным ключом.

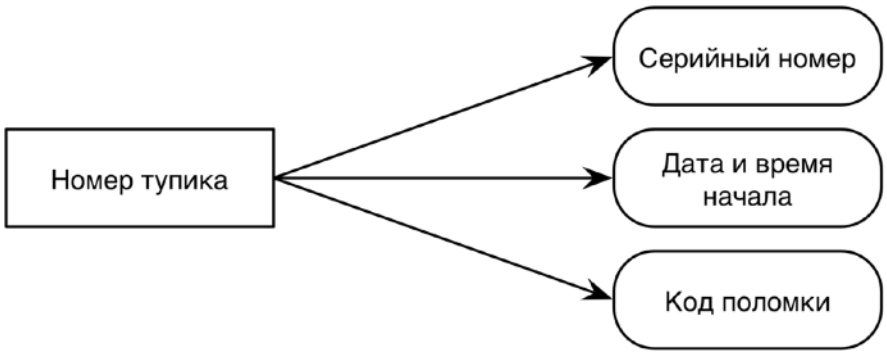


Рисунок 19. Отношение «Ремонтная станция»

В результате проведения даталогического проектирования БД была построена схема данных серверной части ИС (рисунок) для рассмотрения предметной области.

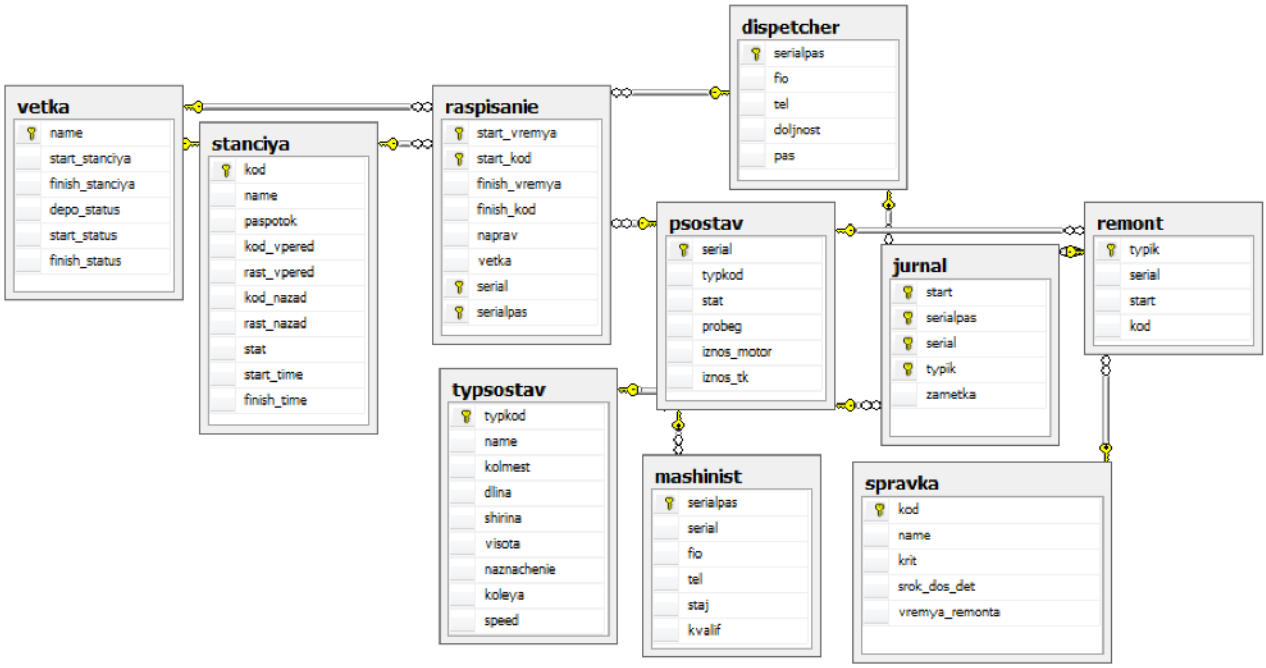


Рисунок 20. Схема данных серверной части ИС

2.3 Разработка основных объектов БД

2.3.1 Задание частных ограничений целостности данных

К частным ограничениям целостности данных относятся пользовательские типы, умолчания и правила. Были разработаны ограничения.

1) Пользовательский тип «Фамилия». Столбцы, содержащие ФИО должны содержать как минимум два слова.

```
CREATE TYPE t_fio
FROM nvarchar(64) not null
GO
CREATE RULE  r_fio
AS
@x like '% %'
GO
exec sp_bindrule  'r_fio', 't_fio'
GO
```

2) Пользовательский тип должность. Допускается использовать только 2 значения: «Старший Диспетчер» и «Диспетчер». По умолчанию используется «Диспетчер».

```
CREATE TYPE t_doljnost
FROM nvarchar(20) not null
GO
CREATE DEFAULT d_doljnost
AS 'Диспетчер'
GO
```

```
EXEC sp_bindefault 'd_doljnost', 't_doljnost'
GO
CREATE RULE r_doljnost
AS @x IN ('Диспетчер', 'Старший Диспетчер')
GO
EXEC sp_bindrule 'r_doljnost', 't_doljnost'
GO
```

3) Правило, задающие ограничение на квалификацию. Допускается использовать тольоко 3 значения «Первая», «Вторая», «Третья».

```
CREATE RULE r_kvalif
AS @x IN ('Первая','Вторая','Третья')
GO
```

4) Значение по умолчанию для столбца Время_открытия таблицы Станция.

```
CREATE DEFAULT d_start_time
AS '5:15'
GO
```

5) Значение по умолчанию для столбца Время_закрытия таблицы Станция.

```
CREATE DEFAULT d_finish_time
AS '23:55'
GO
```

2.3.2 Разработка представлений

Представления делают работу базы данных быстрой и оптимизированной, т.к. представления скрывают от прикладной программы структуру БД, а так же сложность запроса. При создании этих объектов так же обеспечивается защита данных, за счет возможности ограниченного ввода для пользователя.

1) Представление, которое отображает расписание в таком виде, в котором его увидят пассажиры.

```
CREATE VIEW v_raspisanie
AS
SELECT start_vremya AS 'Отправление', s1.name AS 'Со станции',
finish_vremya AS 'Прибытие', s2.name AS 'На станцию', vetka AS 'Ветка'
FROM (raspisanie inner join stanciya s1 on start_kod=kod) inner join stanciya s2
on finish_kod=s2.kod
GO
```

2) Представление для наглядного отображения данных ремонтной станции.

```
CREATE VIEW v_remont
AS
SELECT typik AS 'Тупик', serial AS 'Состав', name AS 'Неисправность', start AS
'Время начала', DATEADD(DAY,srok_dos_det+vremya_remonta,start) AS
'Предположительное окончание'
FROM (remont inner join spravka on remont.kod=spravka.kod)
GO
```

2.3.3 Разработка ХП, функций, триггеров

Разработка хранимых процедур

Хранимые процедуры помогают упростить работу с повторяющимися запросами. Они так же служат для оптимизации работы БД.

1) onesostav. Отвечает за формирование расписания для одного состава. Никаких проверок на корректность данных нет, так как процедура вызывается внутри других процедур, в которых происходит проверка введенных данных.
2) sostav_raspisanie. Формирует расписание для ветки на основе пассажиропотока или количестве составов, веденного диспетчером.
3) prov_raspisanie. Проверка введенной информации перед формированием расписания. Проверяет корректность введенной информации, и выводит информационное сообщение в случае ошибки.
4) alignment_interval. Выравнивает интервалы между составами до равных значений. Используется преимущественно при добавлении и удалении составов. Проверки на введение значения нет, так как процедура вызывается только из других процедур, в которых проверка на корректность ввода предусмотрена.
5) del_sostav_raspisanie. Удаляет состав из сформированного расписания и вызывает ХП выравнивания интервалов между составами в расписании. Если введенного состава нет, выводится сообщение об ошибке.
6) add_sostav_raspis. Добавляет состав в сформированное расписание и вызывает ХП выравнивания интервалов между составами в расписании. Введенные данные проверяются на

ошибки, и в случае их присутствия выводится информационное сообщение.

7) del_rasp. Удаляет расписание для всей ветки. ХП проверяет введенную информацию и выводит сообщение об ошибке, если информация не корректна.

8) add_stanciya. Добавляет станцию в таблицу Станции. ХП проверяет введенную информацию и выводит сообщение об ошибке, если информация не корректна.

9) edit_stanciya. Позволяет редактировать информацию о станции. ХП выполняет проверку введенной информации перед добавлением, и выводит сообщение об ошибке, если введенные данные не соответствуют требованиям.

10) del_stanciya. Позволяет удалять станции из таблицы Станции, если для станции отсутствует расписание. ХП проверяет введенную информацию и выводит сообщение об ошибке, если информация не корректна.

11) add_vetka. Добавление новой ветки в таблицу Ветки. ХП проверяет введенную информацию и выводит сообщение об ошибке, если информация не корректна.

12) del_vetka, Позволяет удалять ветки из таблицы Ветки. ХП проверяет введенную информацию и выводит сообщение об ошибке, если информация не корректна.

13) del_vetka. Добавление состава в таблицу. ХП проверяет введенную информацию и выводит сообщение об ошибке, если информация не корректна.

14) del_sostav. Удаление состава из таблицы Составы. ХП проверяет введенную информацию и выводит сообщение об ошибке, если информация не корректна.

15) edit_psostav. Изменение информации в таблице Составы. ХП проверяет введенную информацию и выводит сообщение об ошибке, если информация не корректна.

16) add_remont. Позволяет отправить состав на техническое обслуживание в ремонтную станцию. ХП проверяет введенную информацию и выводит сообщение об ошибке, если информация не корректна.

17) del_remont. Возвращает состав из ремонтной станции в депо ветки. ХП проверяет введенную информацию и выводит сообщение об ошибке, если информация не корректна.

18) add_typik. Добавляет тупик для таблицы ремонтной станции. ХП проверяет введенную информацию и выводит сообщение об ошибке, если информация не корректна.

19) del_typik. Удаляет тупик из ремонтной станции. ХП проверяет введенную информацию и выводит сообщение об ошибке, если информация не корректна.

20) add_dispetcher. Добавляет диспетчера в таблицу. ХП проверяет введенную информацию и выводит сообщение об ошибке, если информация не корректна.

21) edit_dispetcher. Позволяет изменять информацию о диспетчере в таблице. ХП проверяет введенную информацию и выводит сообщение об ошибке, если информация не корректна.

22) del_dispetcher. Удаляет диспетчера из таблицы. ХП проверяет введенную информацию и выводит сообщение об ошибке, если информация не корректна.

23) add_mashinist. Добавляет машиниста в таблицу. ХП проверяет введенную информацию и выводит сообщение об ошибке, если информация не корректна.

24) edit_mashinist. Позволяет изменять информацию о машинистах в таблице. ХП проверяет введенную информацию и выводит сообщение об ошибке, если информация не корректна.

25) del_mashinist. Удаляет машиниста из таблицы. ХП проверяет введенную информацию и выводит сообщение об ошибке, если информация не корректна.

26) add_typsostav. Добавляет новый тип составов. ХП проверяет введенную информацию и выводит сообщение об ошибке, если информация не корректна.

27) edit_typsostav. Позволяет изменять информацию о типе состава. ХП проверяет введенную информацию и выводит сообщение об ошибке, если информация не корректна.

28) del_typsostav. Удаляет тип составов. ХП проверяет введенную информацию и выводит сообщение об ошибке, если информация не корректна.

Разработка функций пользователя

Функция — фрагмент программного кода (подпрограмма), к которому можно обратиться из другого места программы.

Функция может принимать параметры и должна возвращать некоторое значение, возможно пустое.

1) kol_stanciy. Определяет количество станций на ветке. Исходные данные передаются в виде входных параметров. В качестве результата функция возвращает количество станций на данной ветке.

2) adtime. Определяет время движения от станции до станции. Исходные данные передаются в виде входных параметров. В качестве результата функция возвращает время, необходимо для перемещения на соседнюю станцию.

3) dlina_vetka. Определяет длину ветки. Исходные данные передаются в виде входных параметров. В качестве результата функция возвращает длину данной ветки.

4) dlina. Определяет расстояние между станциями с учетом направления движения. Исходные данные передаются в виде входных параметров. В качестве результата функция возвращает расстояние между двумя любыми станциями.

5) paspotok_vetki. Вычисляет значение пассажиропотока на ветке. Исходные данные передаются в виде входных параметров. В качестве результата функция возвращает общий пассажиропоток на ветке.

6) start_interval. Вычисляет оптимальный интервал между составами. Исходные данные передаются в виде входных параметров. В качестве результата функция возвращает оптимальный интервал между составами.

7) maxi_sostav. Определяет максимальное количество составов, которое допустимо на ветке. Исходные данные передаются в виде входных параметров. В качестве результата функция возвращает максимальное количество составов, которое будет безопасно для данной ветки.

8) proverka_sostav. Проверяет техническое состояние состава. Исходные данные передаются в виде входных параметров. В качестве результата функция возвращает код поломки.sled_sostav. Определение кода состава, который идет перед заданным. Исходные данные передаются в виде входных параметров. В качестве результата функция возвращает значение кода состава, который идет перед данным составом.

Разработка триггеров

Триггеры необходимы для сохранения целостности данных. Они срабатывают автоматически, не требуя от пользователя непосредственного вызова.

1) t_add_stanciya. При добавлении станции в таблицу сохраняет логическую целостность ветки.
2) t_del_typsostav. При удалении типа состава из таблицы поддерживает стратегию каскадирования.
3) t_del_stanciya. При удалении станции сохраняет логическую целостность ветки.

3 Тестирование объектов БД

3.1 Тестирование частных ограничений целостности данных

1) Тестирование правила r_fio.

Тестовый вариант 1.

Исходные данные:

serialpas: 6110546784

fio: Александр

tel: 89102348768

doljnost: Старший Диспетчер

pas: HASHBYTES(‘MD5','6110546784')

Ожидаемый результат: ошибка, данные не будут добавлены.

Тестирование:

INSERT INTO dispetcher

VALUES

(6110546784, 'Александр', '89102348768', 'Старший Диспетчер', HASHBYTES(‘MD5','6110546784'))

Полученный результат: ошибка.

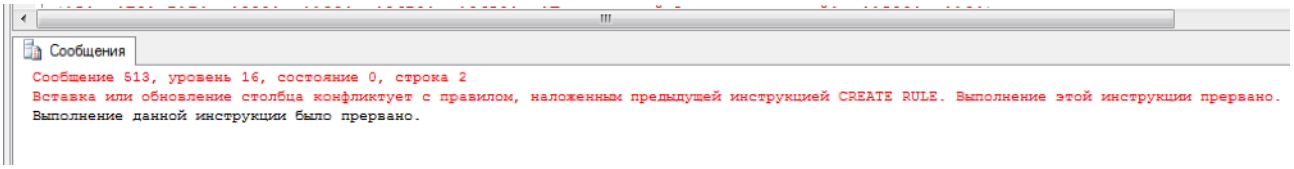


Рисунок 21. Результат тестирования ограничения «r_fio» (тестовый вариант 1)

Тестовый вариант 2.

Исходные данные:

serialpas: 6110546784

fio: Александр Акимов

tel: 89102348768

doljnost: Старший Диспетчер

pas: HASHBYTES(‘MD5’,'6110546784')

Ожидаемый результат: данные будут добавлены.

Тестирование:

INSERT INTO dispetcher

VALUES

(6110546784, 'Александр Акимов', '89102348768', 'Старший Диспетчер', HASHBYTES(‘MD5','6110546784'))

Полученный результат произошло добавление данных

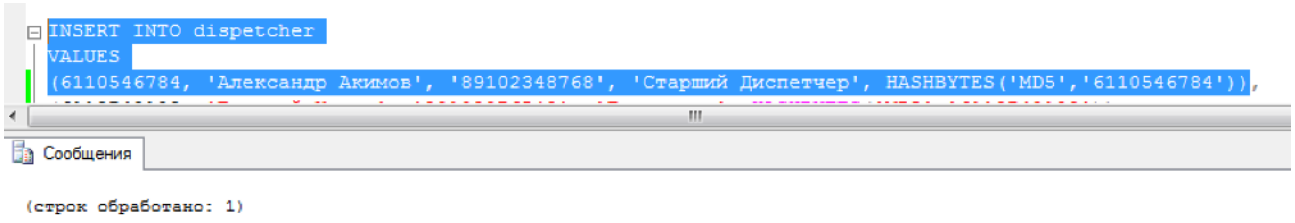


Рисунок 22. Результат тестирования ограничения «r_fio» (тестовый вариант 2)

2) Тестирование правила r_kvalif.

Тестовый вариант 1.

Исходные данные:

serialpas: 6110340432

serial: 1

fio: Павел Дуров

tel: 89495571599

staj: 7

kvalif: Четвертая

pas: HASHBYTES('MD5', '6110340432')

Ожидаемый результат: Ошибка, данные не будут добавлены.

Тестирование

INSERT INTO mashinist

VALUES

(6110340432, '1', 'Павел Дуров', '89495571599', '7', 'Вторая', HASHBYTES('MD5', '6110340432'))

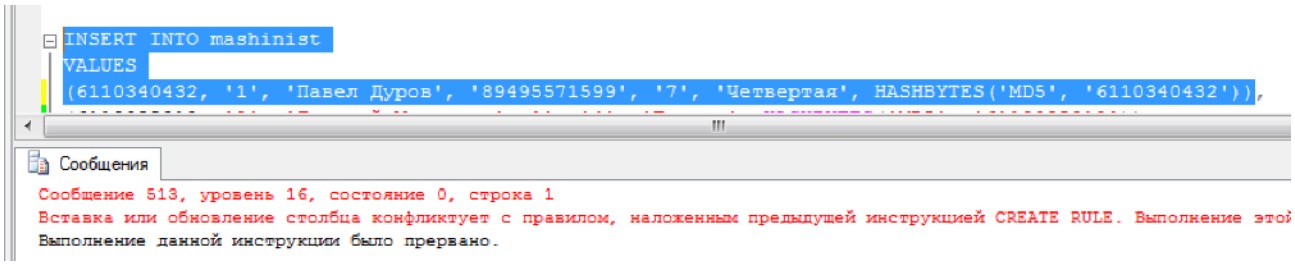


Рисунок 23. Результат тестирования ограничения «r_kvalif» (тестовый вариант 1)

Тестовый вариант 2.

Исходные данные:

serialpas: 6110340432

serial: 1

fio: Павел Дуров

tel: 89495571599

staj: 7

kvalif: Третья

pas: HASHBYTES('MD5', '6110340432')

Ожидаемый результат: Данные будут добавлены.

Тестирование

INSERT INTO mashinist

VALUES (6110340432, '1', 'Павел Дуров', '89495571599', '7', 'Третья', HASHBYTES('MD5', '6110340432'))

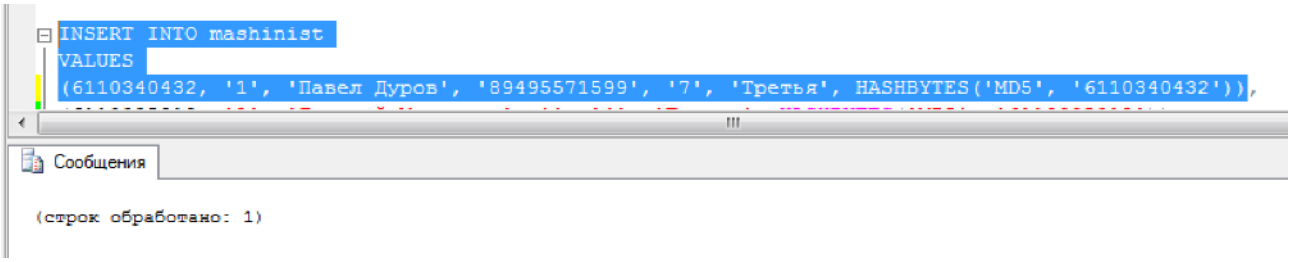


Рисунок 24. Результат тестирования ограничения «r_kvalif» (тестовый вариант 2)

3) Тестирование умолчания «d_start_time».

Тестовый вариант.

Исходные данные:

Не указано время открытия станции.

kod: 1

name: Парк Культуры

paspotok: 49

kod_vpered: 2

rast_vpered: 1000

kod_nazad: NULL

rast_nazad: NULL

stat: 1

finish_time: 22:00

Ожидаемый результат: добавление данных.

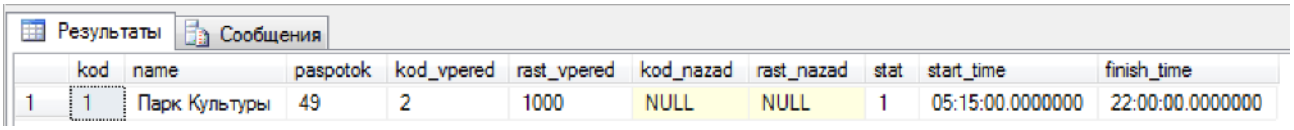
Тестирование:

INSERT INTO stanciya (kod, name, paspotok, kod_vpered, rast_vpered, kod_nazad, rast_nazad, stat, finish_time)

VALUES

('1', 'Парк Культуры', '49', '2', '1000', NULL, NULL, 1, '22:00')

Полученный результат: произошло добавление данных по умолчанию.



Результаты | Сообщения

	kod	name	paspotok	kod_vpered	rast_vpered	kod_nazad	rast_nazad	stat	start_time	finish_time
1	1	Парк Культуры	49	2	1000	NULL	NULL	1	05:15:00.0000000	22:00:00.0000000

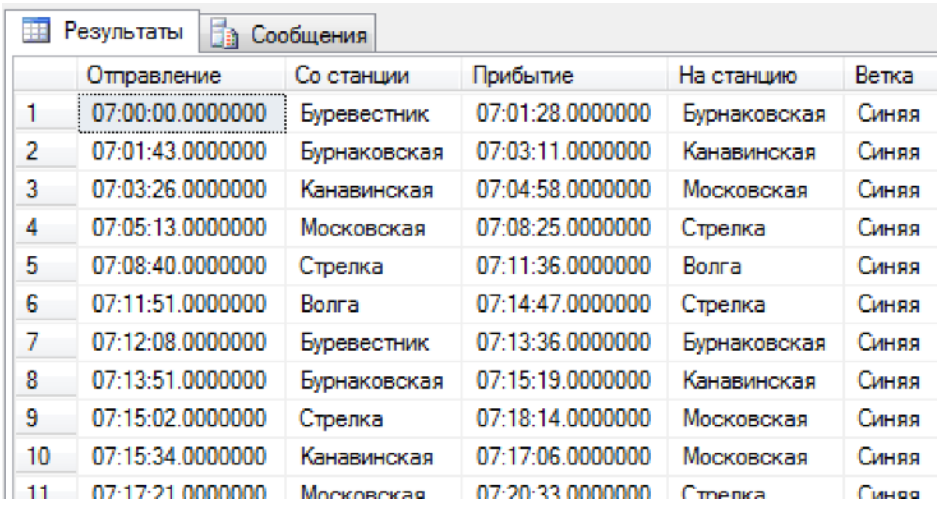
Рисунок 25. Результат тестирования умолчания «d_start_time»

3.2 Тестирование представлений

Для тестирования представлений достаточно проверить корректность структуры выводимых данных.

1) Тестирование представления «v_raspisanie».

select * from v_raspisanie



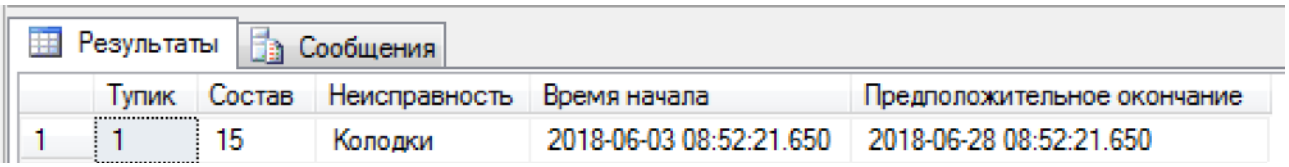
Результаты | Сообщения

	Отправление	Со станции	Прибытие	На станцию	Ветка
1	07:00:00.0000000	Буревестник	07:01:28.0000000	Бурнаковская	Синяя
2	07:01:43.0000000	Бурнаковская	07:03:11.0000000	Канавинская	Синяя
3	07:03:26.0000000	Канавинская	07:04:58.0000000	Московская	Синяя
4	07:05:13.0000000	Московская	07:08:25.0000000	Стрелка	Синяя
5	07:08:40.0000000	Стрелка	07:11:36.0000000	Волга	Синяя
6	07:11:51.0000000	Волга	07:14:47.0000000	Стрелка	Синяя
7	07:12:08.0000000	Буревестник	07:13:36.0000000	Бурнаковская	Синяя
8	07:13:51.0000000	Бурнаковская	07:15:19.0000000	Канавинская	Синяя
9	07:15:02.0000000	Стрелка	07:18:14.0000000	Московская	Синяя
10	07:15:34.0000000	Канавинская	07:17:06.0000000	Московская	Синяя
11	07:17:21.0000000	Московская	07:20:33.0000000	Стрелка	Синяя

Рисунок 26. Результат тестирования представления «v_raspisanie»

2) Тестирование представления «v_remont».

select * from v_remont



	Тупик	Состав	Неисправность	Время начала	Предположительное окончание
1	1	15	Колодки	2018-06-03 08:52:21.650	2018-06-28 08:52:21.650

Рисунок 27. Результат тестирования представления «v_remont»

3.3 Тестирование ХП

Хранимые процедуры и триггеры были протестированы методом черного ящика. Одина из хранимых процедур была протестирована методом белого ящика.

1) Тестирование ХП «add_mashinist» методом черного ящика.

Тестовый вариант 1.

Исходные данные

@serialpas = 340432

@serial = 1123

@fio = ‘Алексей’

@tel = ‘89106578432’

@staj = ‘-2’

@kvalif = ‘Высшая’

Ожидаемый результат: Ошибка.

Тестирование:

EXEC add_mashinist 340432, 1123, 'Алексей', '89106578432', -2, ‘Высшая'

Полученный результат: вывод сообщения «Такой номер паспорта уже есть». Данные в таблицу не добавились.

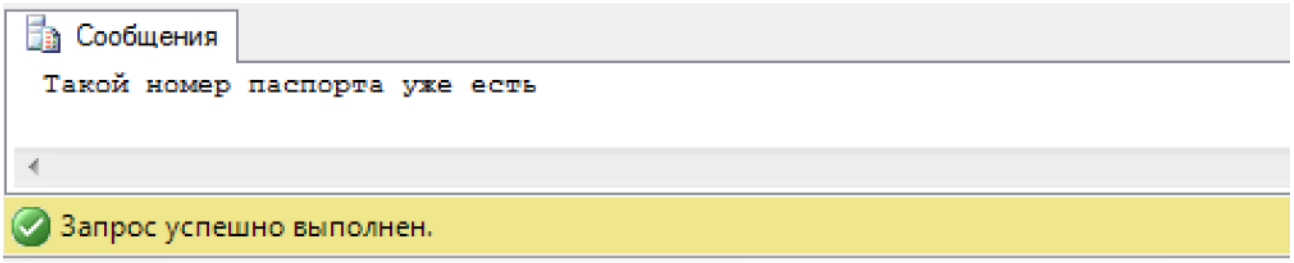


Рисунок 28. Результат тестирования ХП «add_mashinist» методом черного ящика

Тестовый вариант 2.

Исходные данные

@serialpas = 123456

@serial = 1123

@fio = ‘Алексей’

@tel = ‘89106578432’

@staj = ‘-2’

@kvalif = ‘Высшая’

Ожидаемый результат: Ошибка.

Тестирование:

EXEC add_mashinist 123456, 1123, 'Алексей', '89106578432', -2, ‘Высшая'

Полученный результат: вывод сообщения «Состава с таким серийным номером в базе нет!». Данные в таблицу не добавились.

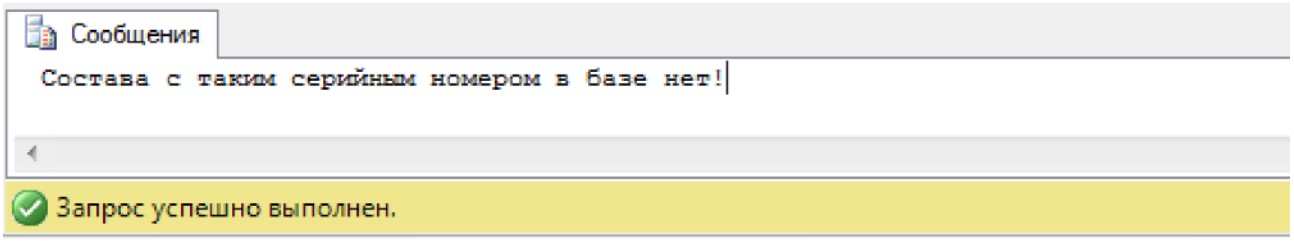


Рисунок 29. Результат тестирования ХП «add_mashinist» методом черного ящика

Тестовый вариант 3.

Исходные данные

@serialpas = 123456

@serial = 1

@fio = ‘Алексей’

@tel = ‘89106578432’

@staj = ‘-2’

@kvalif = ‘Высшая’

Ожидаемый результат: Ошибка.

Тестирование:

EXEC add_mashinist 123456, 1, 'Алексей', '89106578432', -2, ‘Высшая'

Полученный результат: вывод сообщения «Введите коректные значения ФИО». Данные в таблицу не добавились.

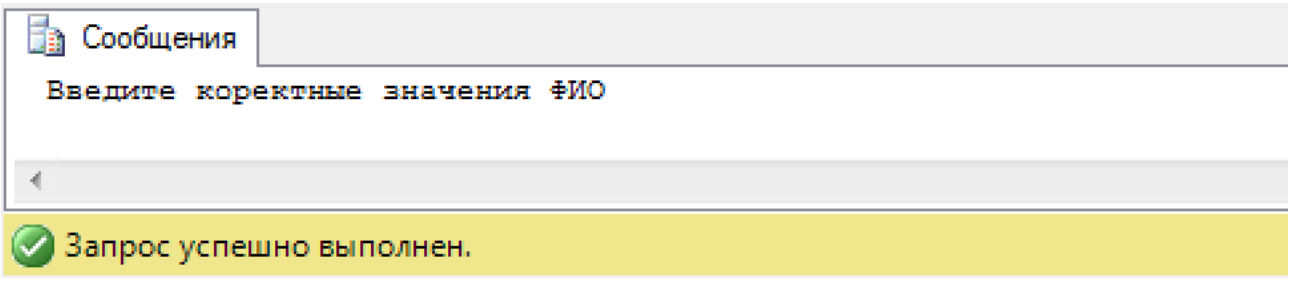


Рисунок 30. Результат тестирования ХП «add_mashinist» методом черного ящика

Тестовый вариант 4.

Исходные данные

@serialpas = 123456

@serial = 1

@fio = ‘Алексей Козлов’

@tel = ‘89106578432’

@staj = ‘-2’

@kvalif = ‘Высшая’

Ожидаемый результат: Ошибка.

Тестирование:

EXEC add_mashinist 123456, 1, ‘Алексей Козлов', '89106578432', -2, ‘Высшая'

Полученный результат: вывод сообщения «Введите корректные данные для квалификации!». Данные в таблицу не добавились.

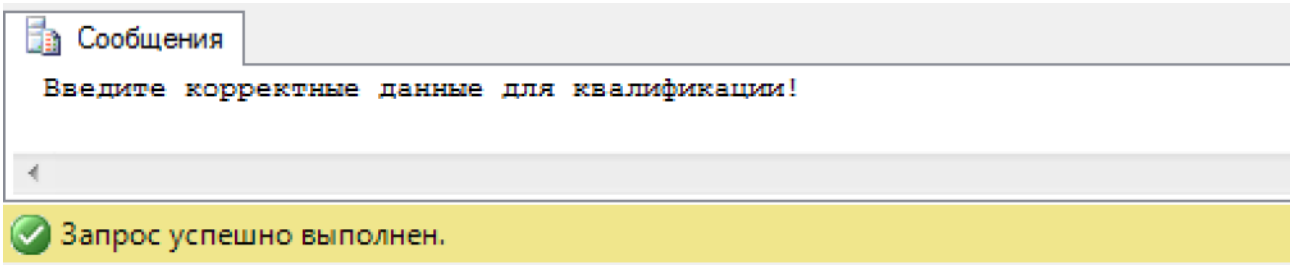


Рисунок 31. Результат тестирования ХП «add_mashinist» методом черного ящика

Тестовый вариант 5.

Исходные данные

@serialpas = 123456

@serial = 1

@fio = ‘Алексей Козлов’

@tel = ‘89106578432’

@staj = ‘-2’

@kvalif = ‘Первая’

Ожидаемый результат: Ошибка.

Тестирование:

EXEC add_mashinist 123456, 1, ‘Алексей Козлов', '89106578432', -2, ‘Первая'

Полученный результат: вывод сообщения «Стаж не может быть отрицательным!». Данные в таблицу не добавились.

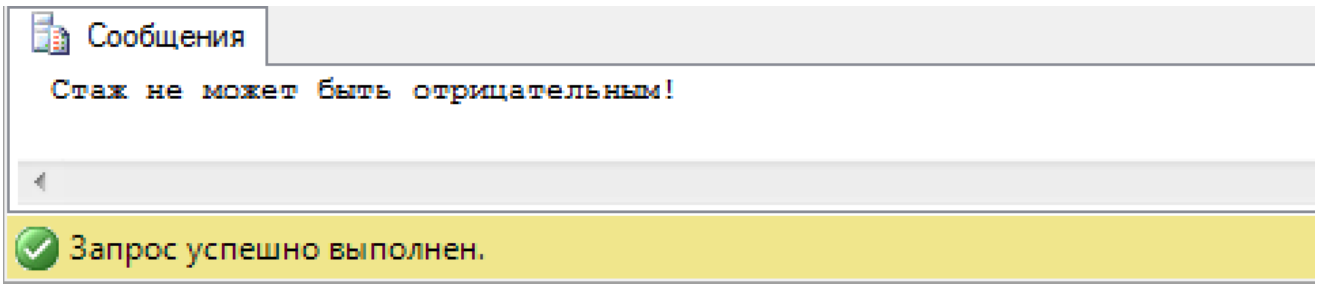


Рисунок 32. Результат тестирования ХП «add_mashinist» методом черного ящика

Тестовый вариант 6.

Исходные данные

@serialpas = 123456

@serial = 1

@fio = ‘Алексей Козлов’

@tel = ‘89106578432’

@staj = ‘5’

@kvalif = ‘Первая’

Ожидаемый результат: Добавление данных.

Тестирование:

EXEC add_mashinist 123456, 1, ‘Алексей Козлов', '89106578432', 5, ‘Первая'

Полученный результат: Данные добавились в таблицу.

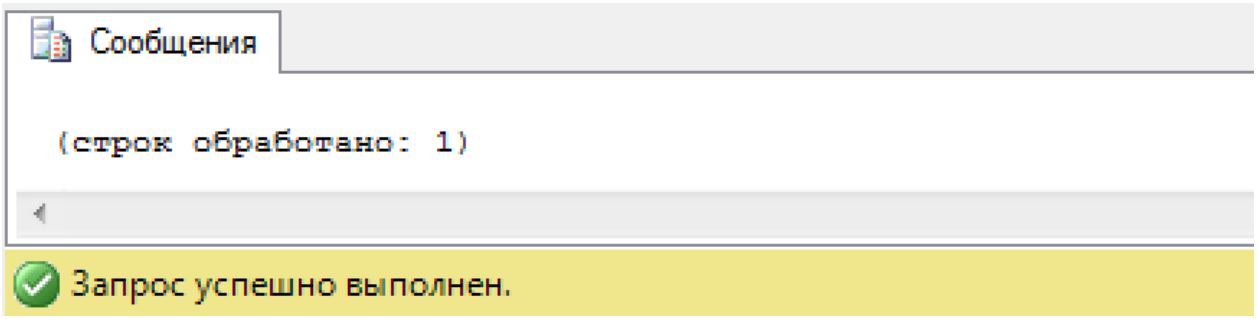


Рисунок 33. Результат тестирования ХП «add_mashinist» методом черного ящика

2) Тестирование ХП «del_stanciya» методом белого ящика:

- Вызов процедуры
- if (станция с веденным кодом существует);
- then if (код введеной станция не участвует в расипсании)
- then удалить станцию
- else вывести сообщение об ошибке
- else вывести сообщение об ошибке
- Конец процедуры

Формирование потокового графа.

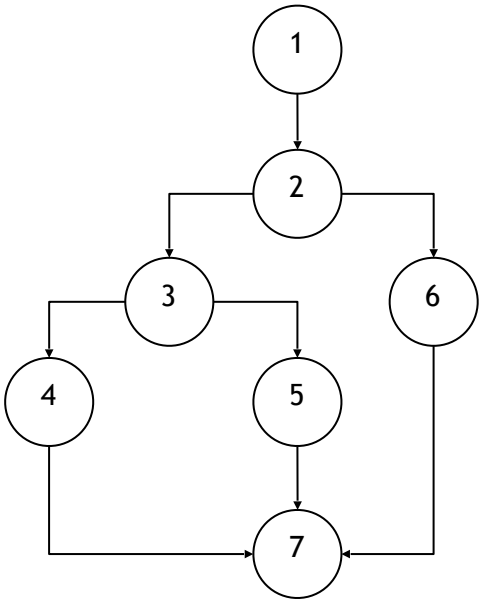


Рисунок 34. Потоковый граф ХП «del_stanciya»

V(G)=3

Базовое множество независимых путей:

1. 1-2-3-4-7;
2. 1-2-3-5-6;
3. 1-2-6-7;

Тестовый вариант 1.

Исходные данные: Существует станция с веденным кодом, и она не участвует в расписании.

Ожидаемый результат: станция удалена, завершение процедуры.

Тестовый вариант 2.

Исходные данные: Существует станция с веденным кодом, но она участвует в расписании.

Ожидаемый результат: Сообщение об ошибке, завершение процедуры.

Тестовый вариант 3.

Исходные данные: Станции с веденным кодом не существует.

Ожидаемый результат: Сообщение об ошибке, завершение процедуры.

Степень покрытия логики процедуры - 100%.

4 Разработка клиентской части ИС

4.1 Выбор режима работы с базой данных, согласно технологии ADO.NET

Клиентское приложение разрабатывалось в среде Microsoft Visual Studio 2017. В качестве языка программирования был выбран язык C#. В клиентском приложении, использующем технологию доступа к БД ADO.NET, используется присоединенный режим.

К основным классам присоединенного режима работы относят:

1. Класс SqlConnection предоставляет соединение с источником данных БД под СУБД MS SQL Server.
2. Класс SqlCommand предназначен для выполнения запросов к БД, которые представляют собой либо SQL-запрос, либо вызов хранимой процедуры.
3. Класс SqlDataReader позволяет просматривать результаты запроса по одной записи за раз. При переходе к следующей записи содержимое предыдущей отбрасывается. Объект SqlDataReader не поддерживает обновление, и возвращаемые им данные доступны только для чтения. Поскольку класс SqlDataReader реализует лишь ограниченный набор функций, он очень прост и имеет высокую производительность. Класс SqlDataReader можно определить как доступный только для чтения последовательный курсор.
4. Класс SqlDataAdapter осуществляет связь между БД и отсоединенными объектами.

Соединение к SQL Server осуществлялся следующем образом.

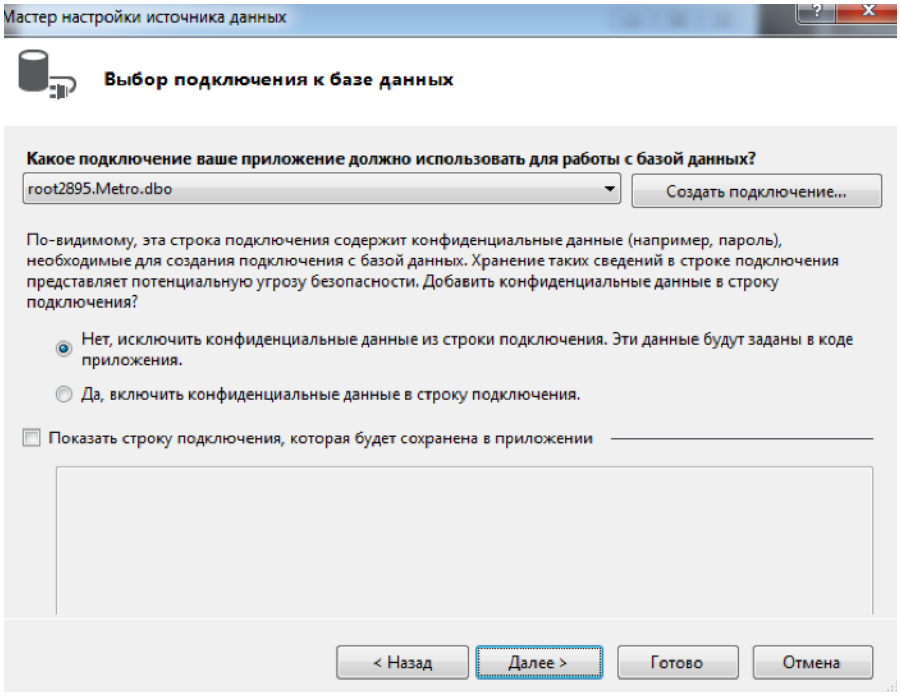


Рисунок 35. Подключение к БД

4.2 Разработка форм

В разработанном приложении содержится 2 формы. Первая форма представляет собой окно входа в программу, включающее в себя авторизацию пользователя. Есть 2 группы пользователей. У каждого пользователя свой пароль.

При неправильном вводе логина или пароля приложение выдает соответствующую ошибку.

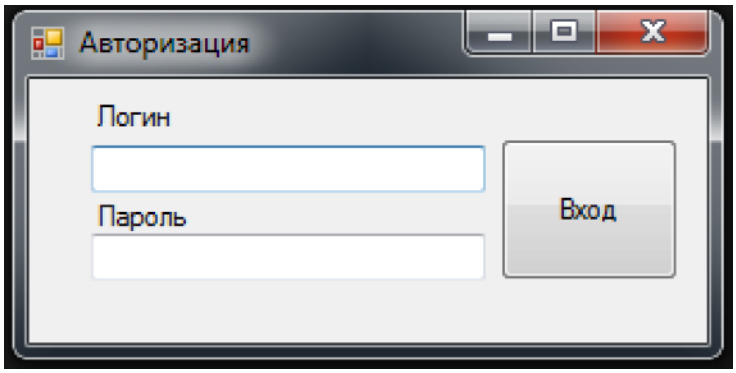


Рисунок 36. Окно авторизации

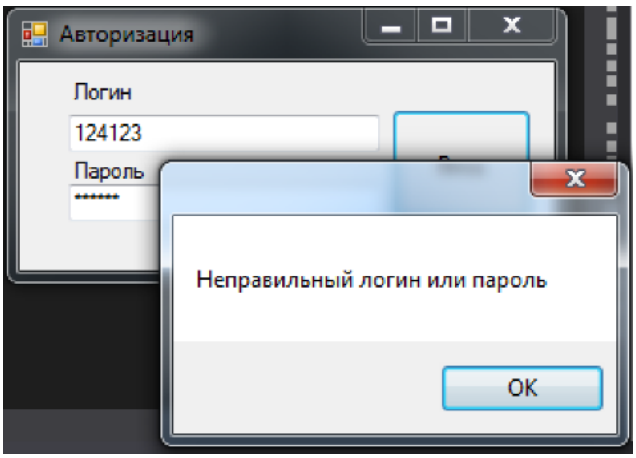


Рисунок 37. Уведомление об ошибочной авторизации

Другая форма предназначена для взаимодействия с функционалом программы. На форме расположен элемент управления TabControl.

Форма служит для выбора вкладок, с которыми пользователь может работать. В зависимости от должности, пользователю выводятся разные элементы управления, необходимые для выполнения его обязанностей.

На каждой вкладке есть кнопки, которые реализуют тот или иной функционал, преимущественно это добавление, удаление или изменение информации. Так же на формах присутствуют поля ввода и элементы DataGridView, предназначенные для вывода информации из БД.

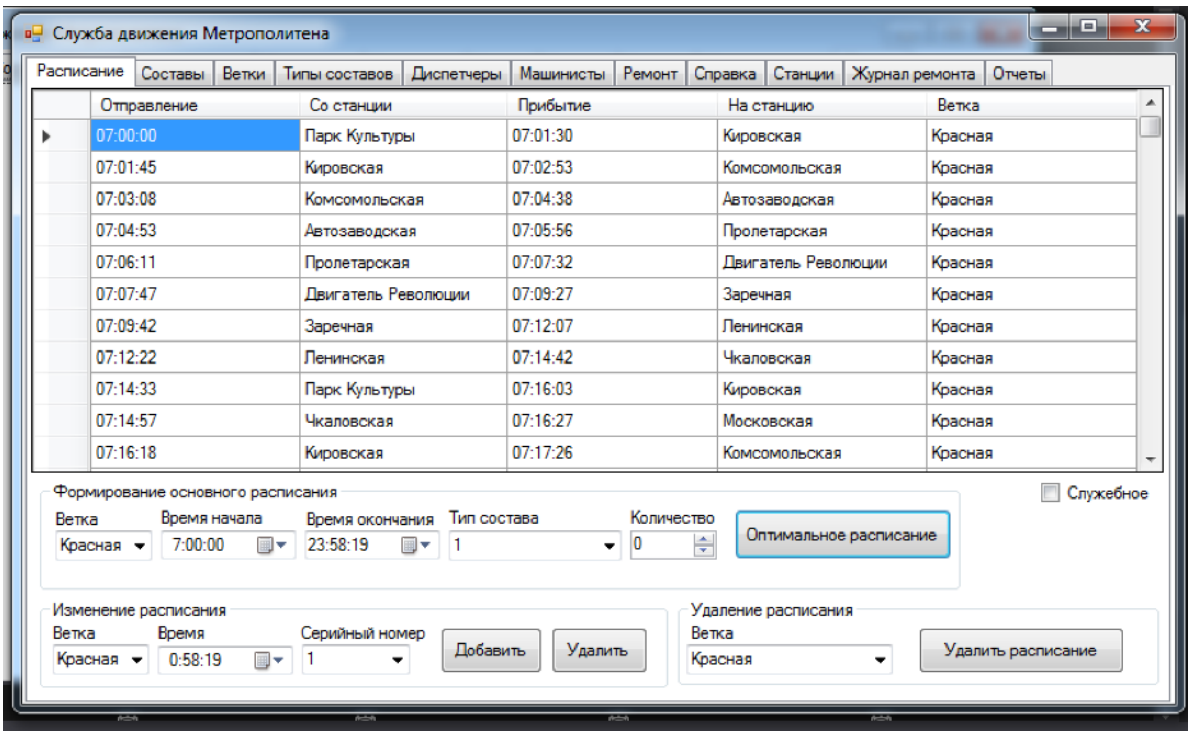


Рисунок 38. Вкладка «Расписание»

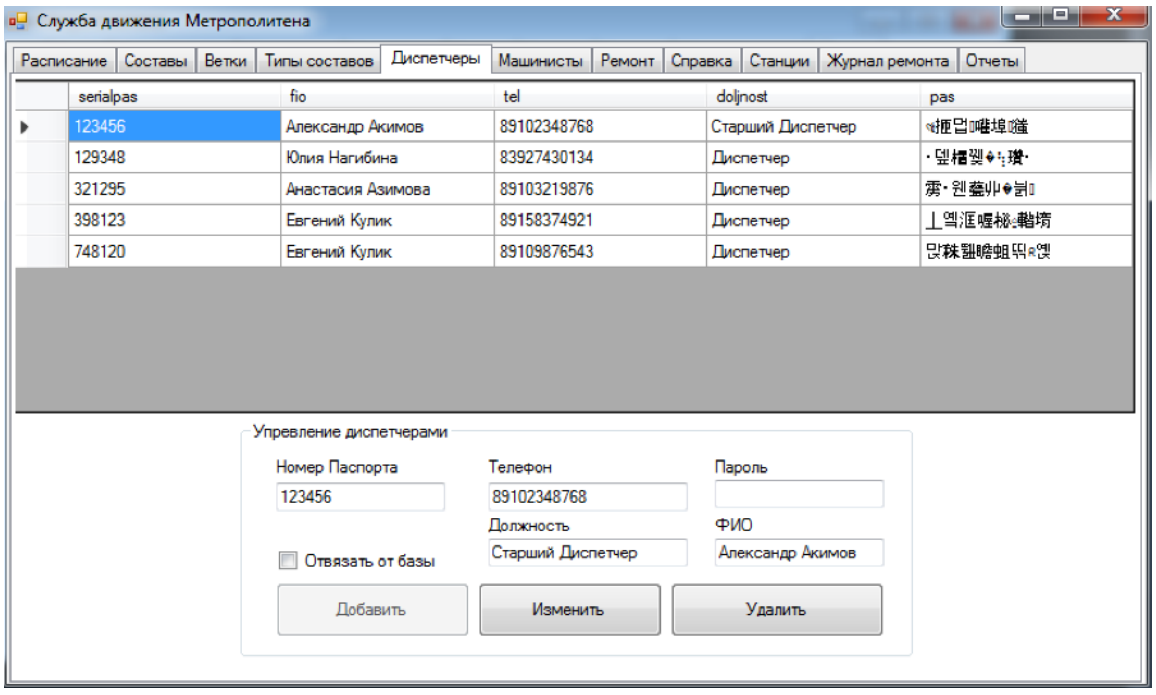


Рисунок 39. Вкладка «Диспетчеры»

Вкладка «Расписание» дает возможность пользователю просматривать информацию о расписании, а также удалить текущее расписание и сформировать новое. Предусмотрена возможность добавления и удаления составов из текущего расписания.

Вкладка «Диспетчеры» дает возможность добавлять, изменять или удалять униформами о сотрудниках, а так же назначать им роли.

5 Разработка программной документации

В соответствии ГОСТ 19.505-79 ЕСПД. Руководство оператора. Требования к содержанию и оформлению (с Изменением N 1).

Назначение программы

Разработанное приложение позволяет упростить автоматизацию деятельности службы движения метрополитена. Оно удобно для пользователя любого уровня. С помощью созданного программного продукта легко сформировать расписание для метрополитена, а так же корректировать его в случае необходимости. Так же приложение предоставляет возможность редактирования других данных посредством добавления, обновления и удаления.

Условия выполнения программы

Для запуска приложения необходимо иметь ПК с минимально допустимыми характеристиками:

- процессор с архитектурой x86-64 (Intel Core i3 3 ГГц и выше)
- оперативная память 2048 Мб и выше;
- жесткий диск 40Гб и выше;
- видеоадаптер (VRAM не менее 512 Мб) или интегрированное графическое ядро процессора (Intel HD Graphics или Radeon R7 series);
- мышка, клавиатура, монитор, принтер.

На данном ПК должна быть установлена лицензионная версия ОС линейки MS Windows. Например, версия: Windows 8 (64-разрядная).

В роли СУБД выступает MS SQL Server 2008.

Выполнение программы

Рассмотрим стандартное обращение диспетчера к приложению - формирование нового оптимального расписания на день.

1) Диспетчеру необходимо сформировать расписание движения на день. Для начала необходимо перейти во вкладку «Расписание», далее заполнить входные параметры и отправить данные в хранимую процедуру, которая в свою очередь сформирует расписание.

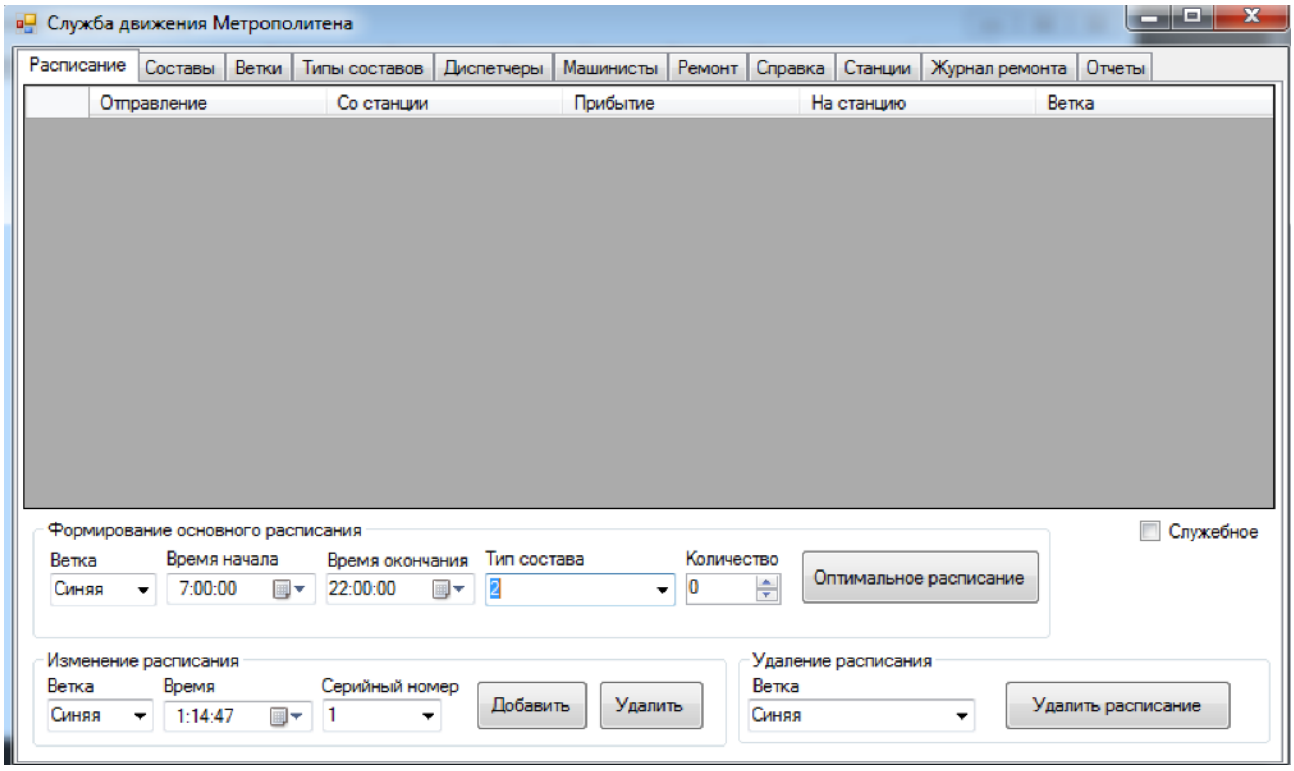


Рисунок 40. Вкладка «Расписание»

Отправление	Со станции	Прибытие	На станцию	Ветка
07:00:00	Буревестник	07:01:28	Бурнаковская	Синяя
07:01:43	Бурнаковская	07:03:11	Канавинская	Синяя
07:03:26	Канавинская	07:04:58	Московская	Синяя
07:05:13	Московская	07:08:25	Стрелка	Синяя
07:08:40	Стрелка	07:11:36	Волга	Синяя
07:11:51	Волга	07:14:47	Стрелка	Синяя
07:12:08	Буревестник	07:13:36	Бурнаковская	Синяя
07:13:51	Бурнаковская	07:15:19	Канавинская	Синяя
07:15:02	Стрелка	07:18:14	Московская	Синяя
07:15:34	Канавинская	07:17:06	Московская	Синяя
07:17:21	Московская	07:20:33	Стрелка	Синяя

Рисунок 41. Результат формирования нового расписания

2) Теперь диспетчер в случае необходимости может добавить новый состав в расписание.

Отправление	Со станции	Прибытие	На станцию	Ветка
07:00:00	Буревестник	07:01:28	Бурнаковская	Синяя
07:01:43	Бурнаковская	07:03:11	Канавинская	Синяя
07:02:00	Буревестник	07:03:28	Бурнаковская	Синяя
07:03:26	Канавинская	07:04:58	Московская	Синяя
07:03:43	Бурнаковская	07:05:11	Канавинская	Синяя
07:05:13	Московская	07:08:25	Стрелка	Синяя
07:05:26	Канавинская	07:06:58	Московская	Синяя
07:07:13	Московская	07:10:25	Стрелка	Синяя
07:08:40	Стрелка	07:11:36	Волга	Синяя
07:10:40	Стрелка	07:13:36	Волга	Синяя
07:11:51	Волга	07:14:47	Стрелка	Синяя

Рисунок 42. Результат добавления нового состава

3) Так же диспетчер может удалить состав из расписания. Для этого необходимо указать серийный номер состава и время удаления.

Отправление	Со станции	Прибытие	На станцию	Ветка
07:02:00	Буревестник	07:03:28	Бурнаковская	Синяя
07:03:43	Бурнаковская	07:05:11	Канавинская	Синяя
07:05:26	Канавинская	07:06:58	Московская	Синяя
07:07:13	Московская	07:10:25	Стрелка	Синяя
07:10:40	Стрелка	07:13:36	Волга	Синяя
07:12:08	Буревестник	07:13:36	Бурнаковская	Синяя
07:13:51	Бурнаковская	07:15:19	Канавинская	Синяя
07:13:51	Волга	07:16:47	Стрелка	Синяя
07:15:34	Канавинская	07:17:06	Московская	Синяя
07:17:02	Стрелка	07:20:14	Московская	Синяя
07:17:21	Московская	07:20:33	Стрелка	Синяя

Рисунок 43. Результат удаления

Аналогично так же операции можно повторить и с другими вкладками. После выполнения всех действий с приложением, нужно нажать на крестик, после подтверждения все окна программы будут закрыты.

Сообщения оператору

В ходе выполнения программы возможен ввод некорректной информации. Тогда появится окно предупреждения. Например, при вводе нового состава в расписание, которое не сформировано для данной ветки, возникает окно подсказки.

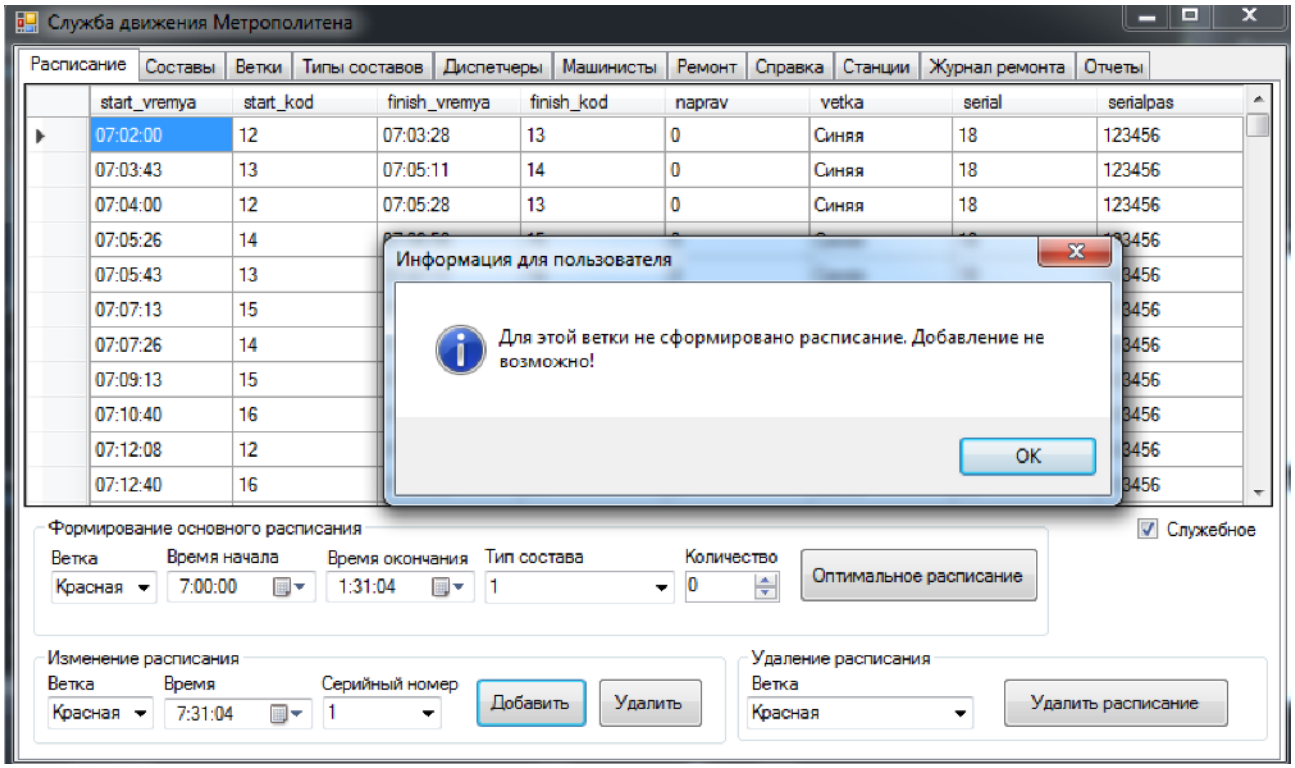


Рисунок 44. Окно подсказки

ЗАКЛЮЧЕНИЕ

В результате выполнения курсового проекта была создана информационная система для службы движения метрополитена, автоматизирующая процессы формирования расписания движения составов.

Проект включил в себя инфологическое и даталогическое проектирования БД. В ходе разработки были созданы и протестированы основные объекты БД.

Так же при написании клиентского приложения, получены практические знания по языку программирования C#. Приобретены навыки в работе в программе Microsoft Visual Studio 2017.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1) Гринченко Н. Н. Конспект лекций по дисциплине «БД и Клиент-серверные приложения» 2017.

2) Введение в базы данных [Электронный ресурс] http://www.mstu.edu.ru/study/materials/zelenkov/toc.html (дата обращения: 25.05.2017)

3) Стиллмен Э., Грин Дж. «Изучаем C#. 3-е изд.»; Издательство Питер; 2014 – 816 с.

Приложение 1. Сценарий создания объектов

Создание таблиц, правил и умолчаний

```
USE master
GO
CREATE DATABASE Metro
ON
(NAME='Metro_Data',
FILENAME='C:\Metro\Metro.mdf',
SIZE=5,
MAXSIZE=100,
FILEGROWTH=2)
LOG ON
(NAME='Metro_Log',
FILENAME='C:\Metro\Metro.ldf',
SIZE=5,
MAXSIZE=100,
FILEGROWTH=2)
GO
USE Metro
GO

CREATE DEFAULT d_start_time
AS '5:15'
GO

CREATE DEFAULT d_finish_time
AS '23:55'
GO

CREATE RULE r_kvalif
```

```
AS @x IN ('Первая','Вторая','Третья')
GO

CREATE TYPE t_doljnost
FROM nvarchar(20) not null
GO
CREATE DEFAULT d_doljnost
AS 'Диспетчер'
GO
EXEC sp_bindefault 'd_doljnost', 't_doljnost'
GO
CREATE RULE r_doljnost
AS @x IN ('Диспетчер', 'Старший Диспетчер')
GO
EXEC sp_bindrule 'r_doljnost', 't_doljnost'
GO

CREATE TYPE t_fio
FROM nvarchar(64) not null
GO
CREATE RULE  r_fio
AS
@x like '% %'
GO
exec sp_bindrule  'r_fio', 't_fio'
GO

CREATE TABLE typsostav (
  typkod int PRIMARY KEY,
  name nvarchar(20),
  kolmest int,
  dlina float CHECK (dlina < 200) not null,
```

```
  shirina int CHECK (shirina < 3000) not null,
  visota int CHECK (visota < 5000) not null,
  naznachenie nvarchar(30),
  koleya int CHECK (koleya=1520) not null,
  speed int
)

CREATE TABLE psostav (
  serial int PRIMARY KEY,
  typkod int,
  stat int,
  probeg int not null,
  iznos_motor int not null,
  iznos_tk int not null,
  constraint FK_typkod_psostav foreign key (typkod) references typsostav
)
GO

CREATE TABLE mashinist (
  serialpas float PRIMARY KEY,
  serial int,
  fio t_fio,
  tel nvarchar(16) not null,
  staj int not null,
  kvalif nvarchar(10),
  constraint FK_serial_mashinist foreign key (serial) references psostav
)
GO
EXEC sp_bindrule 'r_kvalif', 'mashinist.kvalif'
GO

CREATE TABLE dispetcher (
```

```
  serialpas float PRIMARY KEY,
  fio t_fio,
  tel nvarchar(16) not null,
  doljnost t_doljnost,
  pas nvarchar(1000) not null
)
GO

CREATE TABLE stanciya (
  kod int PRIMARY KEY,
  name nvarchar(20),
  paspotok int,
  kod_vpered int,
  rast_vpered int,
  kod_nazad int,
  rast_nazad int,
  stat int,
  start_time time,
  finish_time time
)
GO

EXEC sp_bindefault 'd_start_time', 'stanciya.start_time'
EXEC sp_bindefault 'd_finish_time', 'stanciya.finish_time'


CREATE TABLE vetka (
  name nvarchar(20) PRIMARY KEY,
  start_stanciya int,
  finish_stanciya int,
  depo_status int,
  start_status int,
```

```
  finish_status int,
    constraint FK_start_stanciya foreign key (start_stanciya) references
stanciya(kod),
    constraint FK_finish_stanciya foreign key (finish_stanciya) references
stanciya(kod),
)
GO

CREATE TABLE spravka (
  kod int PRIMARY KEY,
  name nvarchar(30),
  krit float,
  srok_dos_det int,
  vremya_remonta int
)
GO

CREATE TABLE remont (
  typik int PRIMARY KEY,
  serial int,
  start datetime,
  kod int,
  constraint FK_kod_remont foreign key (kod) references spravka,
  constraint FK_serial_remont foreign key (serial) references psostav
)
GO

CREATE TABLE raspisanie (
  start_vremya time,
  start_kod int,
  finish_vremya time,
  finish_kod int,
```

```
  naprav int,
  vetka nvarchar(20),
  serial int,
  serialpas float,
  constraint PK_jurnal_raspisanie PRIMARY KEY
(start_vremya,serialpas,serial,start_kod),
  constraint FK_serial_raspisanie foreign key (serial) references psostav,
  constraint FK_serialpas_raspisanie foreign key (serialpas) references dispetcher,
  constraint FK_start_kod_raspisanie foreign key (start_kod) references
stanciya(kod),
  constraint FK_finish_kod_raspisanie foreign key (finish_kod) references
stanciya(kod),
  constraint FK_vetka_raspisanie foreign key (vetka) references vetka(name)
)
GO

CREATE TABLE jurnal (
  start datetime,
  serialpas float,
  serial int,
  typik int,
  zametka nvarchar(30)
  constraint PK_jurnal PRIMARY KEY (start,serialpas,serial,typik),
  constraint FK_serial_jurnal foreign key (serial) references psostav,
  constraint FK_serialpas_jurnal foreign key (serialpas) references dispetcher,
  constraint FK_typik_jurnal foreign key (typik) references remont
)
GO
```

Создание ХП

```
CREATE PROCEDURE onesostav (@ser int, @vetka varchar(20), @start_time time,
@k_time time, @serialpas float)
AS
BEGIN
  DECLARE  @rab int, @t_vremya time, @typ int, @stanciya int, @next_stanciya int,
@p int, @finish_time time
  SELECT @typ=typkod FROM psostav WHERE serial=@ser
  SET @rab=NULL;
  SELECT @stanciya=start_stanciya FROM vetka WHERE name=@vetka
  SET @p=0
  SET @finish_time=@start_time
  WHILE @finish_time<@k_time
  BEGIN
    IF @p=0
    BEGIN
      SELECT @next_stanciya=kod_vpered FROM stanciya WHERE kod=@stanciya
      IF @next_stanciya is NULL
      BEGIN
        SELECT @next_stanciya=kod_nazad FROM stanciya WHERE kod=@stanciya
        SET @p=1
      END
    END
    ELSE
    BEGIN
      SELECT @next_stanciya=kod_nazad FROM stanciya WHERE kod=@stanciya
      IF (@next_stanciya is NULL)
      BEGIN
        SELECT @next_stanciya=kod_vpered FROM stanciya WHERE kod=@stanciya
        SET @p=0
      END
    END
```

```
   SELECT @finish_time=DATEADD(SECOND,dbo.adtime(@typ, @stanciya,
@p),@start_time)
   IF (SELECT finish_time FROM stanciya WHERE kod=@next_stanciya) >
@finish_time and (SELECT start_time FROM stanciya WHERE kod=@stanciya) <
@start_time
   BEGIN
    IF ((SELECT stat FROM stanciya WHERE kod=@next_stanciya) = 1) or
(@finish_time<@k_time)
     BEGIN
      IF @rab IS NULL
       INSERT INTO raspisanie
        VALUES (@start_time, @stanciya, @finish_time, @next_stanciya, @p,
@vetka, @ser, @serialpas)
      ELSE
      BEGIN
       UPDATE raspisanie
        SET finish_vremya=@finish_time, finish_kod=@next_stanciya, naprav=@p
        WHERE start_vremya=@t_vremya and start_kod=@rab and vetka=@vetka
       SET @rab = NULL
       SET @t_vremya=NULL
      END
     END
     ELSE
     BEGIN
      IF @rab IS NULL
      BEGIN
       INSERT INTO raspisanie (start_vremya, start_kod, vetka, serial, serialpas)
        VALUES (@start_time, @stanciya, @vetka, @ser, @serialpas)
       SET @rab=@stanciya
       SET @t_vremya=@start_time
      END
     END
```

```
    END
    SET @start_time=@finish_time
    SELECT @start_time=DATEADD(SECOND,15,@start_time)
    SET @stanciya=@next_stanciya
  END
  UPDATE psostav
    SET stat = (SELECT start_status FROM vetka WHERE name=@vetka)
    WHERE serial=@ser
END
GO

CREATE PROCEDURE sostav_raspisanie @vetka nvarchar(20), @n_time time,
@k_time time, @kol int, @typ int, @serialpas float
AS
BEGIN
  DECLARE @depo int, @maxi int, @paspotok int, @pasvmestdepo int,
@kol_sostav int, @interval int
  SELECT @depo=depo_status FROM vetka WHERE name=@vetka
  SET @maxi=dbo.maxi_sostav(@vetka, @typ)
  SET @paspotok=dbo.paspotok_vetki(@vetka)
  DECLARE @sostavi_depo TABLE (serial int, stat int, kolmest int, typkod int)
    INSERT INTO @sostavi_depo
    SELECT serial, stat, kolmest, typsostav.typkod
      FROM psostav INNER JOIN typsostav ON psostav.typkod=typsostav.typkod
      WHERE stat=@depo and kolmest!=0 and psostav.typkod=@typ
  DELETE @sostavi_depo
    WHERE dbo.proverka_sostav(serial)!=0 or serial not in (SELECT serial FROM
mashinist)
  SELECT @pasvmestdepo=SUM(kolmest) FROM @sostavi_depo
  IF (@kol=0)
    SET @kol_sostav = CEILING(@paspotok/(SELECT kolmest FROM typsostav
WHERE typkod=@typ))
```

```
  ELSE
    SET @kol_sostav=@kol
  IF (@kol_sostav>@maxi)
  BEGIN
    PRINT 'На ветку будет выведено максимальное количество составов!'
    SET @kol_sostav=@maxi
  END
  SET @interval = dbo.start_interval(@vetka, @typ, @kol_sostav)
  DECLARE @CURSOR CURSOR, @t_ser int, @t_kol int, @k int
  SET @k=0
  SET @CURSOR  = CURSOR SCROLL
  FOR
  SELECT serial, kolmest
    FROM @sostavi_depo
  OPEN @CURSOR
  FETCH NEXT FROM @CURSOR INTO @t_ser, @t_kol
  WHILE (@@FETCH_STATUS = 0 and @k<@kol_sostav)
  BEGIN
    DECLARE @start_time time
    SET @start_time=@n_time
    SELECT @start_time=DATEADD(SECOND,@interval*@k,@n_time)
    EXEC onesostav @t_ser, @vetka, @start_time, @k_time, @serialpas
    SET @k=@k+1
    FETCH NEXT FROM @CURSOR INTO @t_ser, @t_kol
  END
  CLOSE @CURSOR
END
GO

CREATE PROC prov_raspisanie @vetka nvarchar(20), @n_time time, @k_time time,
@kol int, @typ int, @serialpas float
AS
```

```
BEGIN
  IF @vetka in (SELECT name FROM vetka)
    IF NOT EXISTS (SELECT DISTINCT vetka FROM raspisanie WHERE
vetka=@vetka)
      IF (@n_time<@k_time)
        IF (@kol>=0)
          IF @typ in (SELECT typkod FROM typsostav)
            IF EXISTS(SELECT serial FROM psostav WHERE stat=(SELECT
depo_status FROM vetka WHERE name=@vetka) and typkod=@typ)
              EXEC sostav_raspisanie @vetka, @n_time, @k_time, @kol, @typ,
@serialpas
            ELSE PRINT 'Составов данного типа нет в депо этой станции'
          ELSE PRINT 'Такого типа составов не существует'
        ELSE PRINT 'Количество составов не может быть отрицательным'
      ELSE PRINT 'Задайте правильно время'
    ELSE PRINT 'Расписание для этой ветки уже составлено'
  ELSE PRINT 'Такой ветки не существует'
END
GO

CREATE PROCEDURE alignment_interval (@time time, @vetka nvarchar(20), @typ
int, @kol_sostav int)
AS
BEGIN
  DECLARE @CURSOR CURSOR, @t_ser int, @vremya time, @stanciya int
  SELECT @vremya=@time
  SET @CURSOR = CURSOR SCROLL
  FOR
  SELECT DISTINCT serial
    FROM raspisanie
    WHERE start_vremya>=@vremya and vetka=@vetka
  OPEN @CURSOR
```

```
  FETCH NEXT FROM @CURSOR INTO @t_ser
  WHILE @@FETCH_STATUS=0
  BEGIN
    DECLARE @new_interval int, @sled_sostav int, @interval int, @sled_time time,
@prom int
    SET @new_interval=dbo.start_interval(@vetka, @typ, @kol_sostav)
    SET @sled_sostav=dbo.sled_sostav (@t_ser, @vremya)
    SELECT @stanciya=start_kod FROM raspisanie WHERE
start_vremya<DATEADD(SECOND,15,@vremya) and
finish_vremya>DATEADD(SECOND,-15,@vremya) and serial=@sled_sostav
    SELECT TOP 1 @sled_time=start_vremya FROM raspisanie WHERE
serial=@t_ser and start_vremya>@vremya and start_kod=@stanciya ORDER BY
start_vremya ASC
    SET @interval=DATEDIFF(SECOND,@vremya,@sled_time)
    IF (@interval<@new_interval) SET @prom=-1 ELSE SET @prom=1
    DECLARE @CURSOR2 CURSOR, @t_svremya time, @t_fvremya time, @dob int,
@dob2 int
    SET @dob=0 SET @dob2=20
    SET @CURSOR2 = CURSOR SCROLL
    FOR
    SELECT start_vremya, finish_vremya FROM raspisanie WHERE
serial=@sled_sostav and start_vremya>@sled_time ORDER BY start_vremya ASC
    OPEN @CURSOR2
    FETCH NEXT FROM @CURSOR2 INTO @t_svremya, @t_fvremya
    WHILE @@FETCH_STATUS=0
    BEGIN
      IF (@prom=-1)
      BEGIN
        IF(@interval<@new_interval)
        BEGIN
          SET @interval=@interval+20
          SET @dob=@dob+20
```

```
        SET @vremya=DATEADD(SECOND,-@dob+@dob2,@t_svremya)
       END ELSE SET @dob2=0
       UPDATE raspisanie
        SET start_vremya=DATEADD(SECOND,-@dob+@dob2,start_vremya),
finish_vremya=DATEADD(SECOND,-@dob,finish_vremya)
        WHERE start_vremya=@t_svremya and serial=@sled_sostav and
finish_vremya=@t_fvremya
     END
     ELSE
     BEGIN
      IF(@interval>@new_interval)
      BEGIN
        SET @interval=@interval-20
        SET @dob=@dob+20
        SET @vremya=DATEADD(SECOND,@dob+@dob2,@t_svremya)
       END ELSE SET @dob2=0
       UPDATE raspisanie
        SET start_vremya=DATEADD(SECOND,@dob,start_vremya),
finish_vremya=DATEADD(SECOND,@dob-@dob2,finish_vremya)
        WHERE start_vremya=@t_svremya and serial=@sled_sostav and
finish_vremya=@t_fvremya
     END
   FETCH NEXT FROM @CURSOR2 INTO @t_svremya, @t_fvremya
  END
  CLOSE @CURSOR2
  FETCH NEXT FROM @CURSOR INTO @t_ser
  END
  CLOSE @CURSOR
END
GO

CREATE PROCEDURE del_sostav_raspisanie (@ser int, @start_time time)
```

```
AS
BEGIN
  DECLARE @vetka varchar(20), @typ int
  SELECT @typ=typkod FROM psostav WHERE serial=@ser
  SELECT TOP 1 @vetka=vetka FROM raspisanie WHERE serial=@ser
  IF (SELECT stat FROM psostav WHERE serial=@ser) IN (SELECT start_status
FROM vetka WHERE name=@vetka UNION SELECT finish_status FROM vetka
WHERE name=@vetka)
  BEGIN
    DECLARE @kol_sostav int
    SELECT @kol_sostav=COUNT(serial) FROM psostav WHERE stat IN (SELECT
start_status FROM vetka WHERE name=@vetka UNION SELECT finish_status
FROM vetka WHERE name=@vetka)
    SELECT TOP 1 @start_time=start_vremya
      FROM raspisanie
      WHERE start_vremya>=@start_time and start_kod=(SELECT start_stanciya
FROM vetka WHERE name=@vetka) and serial=@ser
      ORDER BY start_vremya ASC
      DELETE FROM raspisanie
        WHERE serial=@ser and start_vremya>=@start_time
      SET @kol_sostav=@kol_sostav-1
      EXEC alignment_interval @start_time, @vetka, @typ, @kol_sostav
    UPDATE psostav
      SET stat = (SELECT depo_status FROM vetka WHERE name=@vetka)
  END
  ELSE PRINT 'Состав не находится на ветке'
END
GO

CREATE PROCEDURE add_sostav_raspis (@ser int, @vetka varchar(20), @time
time, @serialpas float)
AS
```

```
BEGIN
IF EXISTS (SELECT vetka FROM raspisanie WHERE vetka=@vetka)
IF @time> (SELECT MIN(start_vremya) FROM raspisanie) and @time< (SELECT
MAX(finish_vremya) FROM raspisanie)
BEGIN
  DECLARE @typ int, @depo int, @kol_sostav int, @depo_stanciya int, @k_time
time
  SELECT @depo_stanciya=start_stanciya FROM vetka WHERE name=@vetka
  SELECT @typ=typkod FROM psostav WHERE serial=@ser
  SELECT @depo=depo_status FROM vetka WHERE name=@vetka
  SELECT TOP 1 @k_time=start_vremya FROM raspisanie ORDER BY
start_vremya DESC
  SELECT @kol_sostav=COUNT(serial) FROM psostav WHERE stat IN (SELECT
start_status FROM vetka WHERE name=@vetka UNION SELECT finish_status
FROM vetka WHERE name=@vetka)
  IF EXISTS (SELECT serial FROM mashinist WHERE serial=@ser)
    IF (dbo.proverka_sostav(@ser)=0)
      IF (SELECT stat FROM psostav WHERE serial=@ser) = @depo
        IF EXISTS (SELECT * FROM raspisanie)
        BEGIN
          IF NOT EXISTS (SELECT * FROM raspisanie WHERE start_vremya <=
DATEADD(SECOND,90,@time) and start_vremya > DATEADD(SECOND,-90,@time)
and vetka=@vetka and start_kod=@depo_stanciya)
            IF (SELECT DISTINCT typkod FROM psostav WHERE serial = (SELECT
TOP 1 serial FROM raspisanie)) = @typ
              IF ((SELECT COUNT(DISTINCT serial) FROM raspisanie WHERE
vetka=@vetka) < dbo.maxi_sostav(@vetka, @typ))
              BEGIN
                EXEC onesostav @ser, @vetka, @time, @k_time, @serialpas
                SET @kol_sostav=@kol_sostav+1
                EXEC alignment_interval @time, @vetka, @typ, @kol_sostav
              END
```

```
            ELSE PRINT 'На ветке находится макисмально допустимое
количество составов'
          ELSE PRINT 'Тип состава не соответствует типу составов, которые
находятся на ветке'
        ELSE PRINT 'Нельзя пустить состав в это время, выберете другое'
      END
      ELSE EXEC onesostav @ser, @vetka, @time, @k_time, @serialpas
    ELSE PRINT 'Состав не находится в депо ветки, переведите состав в
депо'
  ELSE PRINT 'Состав нуждается в техническом обслуживании'
 ELSE PRINT 'На данный состав не назначен машинист'
END
ELSE PRINT 'Нельзя добавить состав в это время'
ELSE PRINT 'Для этой ветки не сформировано расписание. Добавление не
возможно!'
END
GO

CREATE PROC del_rasp @vetka nvarchar(20)
AS
BEGIN
 IF @vetka in (SELECT name FROM vetka)
  IF EXISTS (SELECT vetka FROM raspisanie WHERE vetka=@vetka)
  BEGIN
    UPDATE psostav
      SET stat=(SELECT depo_status FROM vetka WHERE name=@vetka)
      WHERE serial in (SELECT DISTINCT serial FROM raspisanie WHERE
vetka=@vetka)
  DELETE raspisanie
    WHERE vetka=@vetka
  END
  ELSE PRINT 'Для этой ветки нет расписания'
```

```
  ELSE PRINT 'Такой ветки не существует'
END
GO

CREATE PROCEDURE add_stanciya (@kod int, @name nvarchar(20), @paspotok
int, @kod_nazad int, @rast_nazad int, @rast_vpered int, @stat int, @start_time time,
@finish_time time)
AS
BEGIN
IF (@kod!=0)
  IF NOT EXISTS (SELECT * FROM stanciya WHERE @kod=kod)
    IF (EXISTS (SELECT * FROM stanciya WHERE @kod_nazad=kod) or
@kod_nazad IS NULL or @kod_nazad=0)
      IF (@kod_nazad is not NULL or @kod_nazad!=0) and ((SELECT kod_vpered
FROM stanciya WHERE kod=@kod_nazad) is not NULL) and (@rast_vpered is null
or @rast_vpered=0)
        PRINT 'Невозможно добавить станцию без расстояния вперед, если она
не конечная.'
      ELSE
        IF (@start_time<@finish_time)
          IF (@kod_nazad is not null and @kod_nazad!=0) and (@rast_nazad=0 or
@rast_nazad is null)
            PRINT 'Добавльте расстояние назад'
          ELSE
            INSERT INTO stanciya
              VALUES (@kod, @name, @paspotok, NULL, @rast_vpered, @kod_nazad,
@rast_nazad, @stat, @start_time, @finish_time)
        ELSE PRINT 'Исправьте время открытия/закрытия станции'
    ELSE PRINT 'Не существует станции с кодом: ' + cast(@kod_nazad as
nvarchar(10))
  ELSE PRINT 'Станция с таким кодом уже существует'
ELSE PRINT 'Код станции не может быть 0'
```

```
END
GO

CREATE PROCEDURE edit_stanciya (@kod int, @name nvarchar(20), @paspotok
int, @rast_nazad int, @rast_vpered int, @stat int, @start_time time, @finish_time
time)
AS
BEGIN
  IF EXISTS (SELECT kod FROM stanciya WHERE kod=@kod)
    IF @paspotok>=0
      IF @rast_nazad>=0 or @rast_nazad is NULL
        IF @rast_vpered>=0 or @rast_vpered is NULL
          IF (@finish_time>@start_time)
            IF (SELECT kod_nazad FROM stanciya WHERE kod=@kod) is not null and
(@rast_nazad>0 or @rast_nazad is not null)
            BEGIN
              UPDATE stanciya
                SET paspotok=@paspotok, name=@name, rast_nazad=@rast_nazad,
rast_vpered=@rast_vpered, stat=@stat, start_time=@start_time,
finish_time=@finish_time
                WHERE kod=@kod
              UPDATE stanciya
                SET rast_vpered=@rast_nazad
                WHERE kod=(SELECT kod_nazad FROM stanciya WHERE kod=@kod)
              UPDATE stanciya
                SET rast_nazad=@rast_vpered
                WHERE kod=(SELECT kod_vpered FROM stanciya WHERE kod=@kod)
            END
            ELSE PRINT 'Нельзя добавить расстояние назад'
          ELSE PRINT 'Некорректное время работы станции'
        ELSE PRINT ‘Расстояние вперед не может быть отрицательным'
      ELSE PRINT 'Расстояние назад не может быть отрицательным'
```

```
   ELSE PRINT 'Пассажиропоток не может быть отрицательным'
  ELSE PRINT 'Станции с таким кодом не существует'
END
GO

CREATE PROC del_stanciya (@kod int)
AS
BEGIN
  IF EXISTS (SELECT kod FROM stanciya WHERE kod=@kod)
    IF NOT EXISTS (SELECT start_kod FROM raspisanie WHERE start_kod=@kod)
      DELETE stanciya
        WHERE kod=@kod
    ELSE PRINT 'Нельзя удалить станцию, пока она в расписании'
  ELSE PRINT 'Станции с таким кодом нет'
END
GO

CREATE PROCEDURE add_vetka (@name varchar(20), @start_stanciya int,
@depo_status int, @start_status int, @finish_status int)
AS
BEGIN
  IF NOT EXISTS (SELECT name FROM vetka WHERE name=@name)
    IF @start_stanciya not in (SELECT start_stanciya FROM vetka) and (SELECT
kod_vpered FROM stanciya WHERE kod=@start_stanciya) is NOT NULL
      IF EXISTS (SELECT * FROM stanciya WHERE kod=@start_stanciya)
        IF ((SELECT kod_nazad FROM stanciya WHERE kod=@start_stanciya) is null)
          IF @depo_status NOT IN (SELECT depo_status FROM vetka)
            IF @start_status NOT IN (SELECT start_status FROM vetka)
              IF @finish_status NOT IN (SELECT finish_status FROM vetka)
                IF @finish_status!=@start_status and @start_status!=@depo_status and
@depo_status!=@finish_status
                  BEGIN
```

```
            DECLARE @finish_stanciya int
            SELECT @finish_stanciya=kod_vpered FROM stanciya WHERE
kod=@start_stanciya
            WHILE (SELECT kod_vpered FROM stanciya WHERE
kod=@finish_stanciya) is NOT NULL
              SELECT @finish_stanciya=kod_vpered FROM stanciya WHERE
kod=@finish_stanciya
            INSERT INTO vetka (name, start_stanciya, finish_stanciya, depo_status,
start_status, finish_status)
            VALUES (@name, @start_stanciya, @finish_stanciya, @depo_status,
@start_status, @finish_status)
          END
          ELSE PRINT 'Значения статусов должны быть отличными друг от
друга'
        ELSE PRINT 'Выберете другой статус движения состава к депо'
      ELSE PRINT 'Выберете другой статус движения состава от депо'
    ELSE PRINT 'Выберете другой идентификатор статуса депо'
  ELSE PRINT 'Станция не может быть депо. Выберете или создайте
другую станцию'
 ELSE PRINT 'Станции для депо нет, создайте станцию депо'
ELSE PRINT 'Выберете другую станцию для депо'
ELSE PRINT 'Такая ветка уже существует'
END
GO

CREATE PROC del_vetka (@name nvarchar(20), @name2 nvarchar(20))
AS
BEGIN
 IF EXISTS (SELECT name FROM vetka WHERE name=@name)
 BEGIN
  DECLARE @depo int, @depo2 int
  SELECT @depo=depo_status FROM vetka WHERE name=@name
```

```
    SELECT @depo2=depo_status FROM vetka WHERE name=@name2
    IF @name in (SELECT vetka FROM raspisanie WHERE vetka=@name)
      EXEC del_rasp @name
    IF EXISTS (SELECT serial FROM psostav WHERE typkod=@depo) and @name2
is not null
      UPDATE psostav SET stat = @depo2 WHERE stat = @depo
    ELSE
      UPDATE psostav SET stat = NULL WHERE stat = @depo
    DELETE vetka
      WHERE name=@name
  END
  ELSE PRINT 'Ветки с таким названием не сущетсвует'
END
GO

CREATE PROC del_vetka (@name nvarchar(20), @name2 nvarchar(20))
AS
BEGIN
  IF EXISTS (SELECT name FROM vetka WHERE name=@name)
  BEGIN
    DECLARE @depo int, @depo2 int
    SELECT @depo=depo_status FROM vetka WHERE name=@name
    SELECT @depo2=depo_status FROM vetka WHERE name=@name2
    IF @name in (SELECT vetka FROM raspisanie WHERE vetka=@name)
      EXEC del_rasp @name
    IF EXISTS (SELECT serial FROM psostav WHERE typkod=@depo) and @name2
is not null
      UPDATE psostav SET stat = @depo2 WHERE stat = @depo
    ELSE
      UPDATE psostav SET stat = NULL WHERE stat = @depo
    DELETE vetka
      WHERE name=@name
```

```
  END
  ELSE PRINT 'Ветки с таким названием не сущетсвует'
END
GO

CREATE PROCEDURE del_sostav (@serial int)
AS
BEGIN
  IF EXISTS (SELECT serial FROM psostav WHERE serial=@serial)
    IF NOT EXISTS (SELECT serial FROM raspisanie WHERE serial=@serial)
    BEGIN
      IF EXISTS (SELECT serial FROM remont WHERE serial=@serial)
        UPDATE remont
          SET serial=NULL, start=NULL, kod=NULL
          WHERE serial=@serial
      IF EXISTS (SELECT serial FROM mashinist WHERE serial=@serial)
        UPDATE mashinist
          SET serial=NULL
          WHERE serial=@serial
      DELETE psostav
        WHERE serial=@serial
    END
    ELSE PRINT 'Невозможно удалить состав, так как он числится в
расписании'
  ELSE PRINT 'Состава с таким серийным номером нет'
END
GO

CREATE PROC edit_psostav (@serial int, @typkod int, @vetka nvarchar(20),
@probeg float, @iznos_motor float, @iznos_tk float)
AS
BEGIN
```

```
  IF EXISTS (SELECT name FROM vetka WHERE name=@vetka)
    IF (@probeg>=0 and @iznos_motor>=0 and @iznos_tk>=0)
      IF EXISTS (SELECT * FROM psostav WHERE serial=@serial)
        IF @typkod IN (SELECT typkod FROM typsostav)
          IF ((SELECT stat FROM psostav WHERE serial=@serial) in (SELECT
depo_status FROM vetka)) or (SELECT stat FROM psostav WHERE serial=@serial)
is NULL
            UPDATE psostav
              SET typkod=@typkod, stat=(SELECT depo_status FROM vetka WHERE
name=@vetka), probeg=@probeg, iznos_motor=@iznos_motor, iznos_tk=@iznos_tk
              WHERE serial=@serial
          ELSE PRINT 'Нельзя поменять статус данного состава'
        ELSE PRINT 'Такого типа состава не существует'
      ELSE PRINT 'Такого серийного номера не существует'
    ELSE PRINT 'Введите корректные данные'
  ELSE PRINT 'Такой ветки не существет'
END
GO

CREATE PROCEDURE add_remont (@typik int, @serial int, @date datetime,
@serialpass float, @zametka nvarchar(100))
AS
BEGIN
  IF EXISTS (SELECT typik FROM remont WHERE typik=@typik)
    IF EXISTS (SELECT serial FROM psostav WHERE serial=@serial)
    BEGIN
      DECLARE @kod int
      SET @kod=dbo.proverka_sostav(@serial)
      IF (@kod!=0)
        IF (SELECT stat FROM psostav WHERE serial=@serial) IN (SELECT
depo_status FROM vetka)
```

```
        IF EXISTS (SELECT typik FROM remont WHERE serial IS NULL and
typik=@typik)
        BEGIN
          UPDATE remont
            SET kod=@kod, serial=@serial, start=@date
            WHERE typik=@typik
          INSERT INTO jurnal
            VALUES  (@date, @serialpass, @serial, @typik, @zametka)
          UPDATE psostav SET stat=0 WHERE serial=@serial
        END
        ELSE PRINT 'Этот тупик уже занят!'
       ELSE PRINT 'Состав находится не в депо. Сначала верните его в депо'
     ELSE PRINT 'Состав не нуждается в техническом обслуживании'
   END
   ELSE PRINT 'Такого состава не существует'
 ELSE PRINT 'Такого тупика не существует. Выберете другой.'
END
GO

CREATE PROC del_remont (@serial int, @vetka nvarchar(20), @serialpass float)
AS
BEGIN
 IF EXISTS (SELECT serial FROM psostav WHERE serial=@serial)
   IF EXISTS (SELECT name FROM vetka WHERE name=@vetka)
     IF EXISTS (SELECT serialpas FROM dispetcher WHERE serialpas=@serialpass)
       IF EXISTS (SELECT serial FROM remont WHERE serial=@serial)
       BEGIN
         UPDATE remont
           SET serial=NULL, start=NULL, kod=NULL
           WHERE serial=@serial
         UPDATE psostav
           SET stat=(SELECT depo_status FROM vetka WHERE name=@vetka)
```

```
          WHERE serial=@serial
        END
        ELSE PRINT 'Данный состав не находится на ремонтной станции'
      ELSE PRINT 'Такого диспетчера не существует'
    ELSE PRINT 'Такой ветки не существует'
  ELSE PRINT 'Такого состава не существует'
END
GO

CREATE PROC add_typik (@typik int)
AS
BEGIN
  IF NOT EXISTS (SELECT typik FROM remont WHERE typik=@typik)
    INSERT INTO remont
      VALUES (@typik, NULL, NULL, NULL)
  ELSE PRINT 'Тупик с таким номером уже существует'
END
GO

CREATE PROC del_typik (@typik int)
AS
BEGIN
  IF EXISTS (SELECT typik FROM remont WHERE typik=@typik)
    IF EXISTS (SELECT typik FROM remont WHERE serial is NULL and
typik=@typik)
      DELETE remont
        WHERE typik=@typik
    ELSE PRINT 'Тупик не пуст, переведите состав в депо'
  ELSE PRINT 'Тупика с таким кодом нет'
END
GO
```

```
CREATE PROCEDURE add_dispetcher (@serialpas float, @fio t_fio, @tel
nvarchar(16), @doljnost nvarchar(20), @pas varchar(1000))
AS
BEGIN
  IF NOT EXISTS (SELECT serialpas FROM dispetcher WHERE
serialpas=@serialpas)
    IF (@fio like '% %')
      IF @doljnost in ('Стажер', 'Диспетчер', 'Старший диспетчер')
        INSERT INTO dispetcher
          VALUES (@serialpas, @fio, @tel, @doljnost, HASHBYTES('MD5', @pas))
      ELSE PRINT 'Введите корректную должность!'
    ELSE PRINT 'Введите корректные значения ФИО'
  ELSE PRINT 'Такой номер паспорта уже есть в базе!'
END
GO

CREATE PROC edit_dispetcher (@serialpas float, @fio t_fio, @tel nvarchar(16),
@doljnost nvarchar(20), @pas varchar(1000))
AS
BEGIN
  IF EXISTS (SELECT serialpas FROM dispetcher WHERE serialpas=@serialpas)
    IF @doljnost in ('Стажер', 'Диспетчер', 'Старший диспетчер')
      IF (@fio like '% %')
      BEGIN
        UPDATE dispetcher
          SET fio=@fio, tel=@tel, doljnost=@doljnost
          WHERE serialpas=@serialpas
        IF (@pas IS NOT NULL)
          UPDATE dispetcher
            SET pas=HASHBYTES('MD5',@pas)
            WHERE serialpas=@serialpas
      END
```

```
    ELSE PRINT 'Введите корректные значения ФИО'
  ELSE PRINT 'Введите корректную должность!'
 ELSE PRINT 'Диспетчера с такими паспортными данными не существует'
END
GO

CREATE PROC del_dispetcher (@serialpas float)
AS
BEGIN
 IF EXISTS (SELECT serialpas FROM dispetcher WHERE serialpas=@serialpas)
  DELETE dispetcher
   WHERE serialpas=@serialpas
 ELSE PRINT 'Диспетчера с такими пасортными данными нет в базе'
END
GO

CREATE PROCEDURE add_mashinist @serialpas float, @serial int, @fio
nvarchar(20), @tel nvarchar(16), @staj int, @kvalif nvarchar(10)
AS
BEGIN
 IF NOT EXISTS (SELECT serialpas FROM mashinist WHERE
serialpas=@serialpas)
  IF EXISTS (SELECT serial FROM psostav WHERE serial=@serial)
   IF (@fio like '% %')
    IF @kvalif in ('Первая', 'Вторая', 'Третья')
     IF (@staj>=0)
      INSERT INTO mashinist
       VALUES (@serialpas, @serial, @fio, @tel, @staj, @kvalif)
     ELSE PRINT 'Стаж не может быть отрицательным!'
    ELSE PRINT 'Введите корректные данные для квалификации!'
   ELSE PRINT 'Введите коректные значения ФИО'
  ELSE PRINT 'Состава с таким серийным номером в базе нет!'
```

```
  ELSE PRINT 'Такой номер паспорта уже есть'
END
GO

CREATE PROC edit_mashinist (@serialpas float, @serial int, @fio t_fio, @tel
nvarchar(16), @staj int, @kvalif nvarchar(10))
AS
BEGIN
  IF EXISTS (SELECT serialpas FROM mashinist WHERE serialpas=@serialpas)
    IF @serial IN (SELECT serial FROM psostav)
      IF @kvalif in ('Первая', 'Вторая', 'Третья')
        IF (@staj>=0)
          UPDATE mashinist
            SET serial=@serial, fio=@fio, tel=@tel, staj=@staj, kvalif=@kvalif
            WHERE serialpas=@serialpas
        ELSE PRINT 'Стаж не может быть отрицательным!'
      ELSE PRINT 'Введите корректные данные для квалификации!'
    ELSE PRINT 'Состава, которым управляет машинист, не существует'
  ELSE PRINT 'Машиниста с такими паспортами данными нет в базе'
END
GO

CREATE PROC del_mashinist (@serialpas float)
AS
BEGIN
  IF EXISTS (SELECT serialpas FROM mashinist WHERE serialpas=@serialpas)
    IF ((SELECT stat FROM psostav WHERE serial=(SELECT serial FROM mashinist
WHERE serialpas=@serialpas)) IN (SELECT depo_status FROM vetka) or EXISTS
(SELECT serial FROM mashinist WHERE serialpas!=@serialpas and
serial=(SELECT serial FROM mashinist WHERE serialpas=@serialpas)))
      DELETE mashinist
        WHERE serialpas=@serialpas
```

```
   ELSE PRINT 'Переведите состав в депо станции, перед удалением
машиниста'
  ELSE PRINT 'Машиниста с такими паспортными данными нет в базе!'
END
GO

CREATE PROCEDURE add_typsostav (@typkod int, @name nvarchar(20),
@kolmest int, @dlina float, @shirina float, @visota float, @koleya int, @naznachenie
nvarchar(20), @speed int)
AS
BEGIN
  IF NOT EXISTS (SELECT typkod FROM typsostav WHERE typkod=@typkod)
    IF (@dlina>0 and @shirina>0 and @visota>0 and @koleya>0 and @kolmest>0)
      IF @dlina < 200
        IF @shirina < 3000
          IF @visota < 5000
            IF @koleya=1520
              IF @speed>0
                INSERT INTO typsostav
                  VALUES (@typkod, @name, @kolmest, @dlina, @shirina, @visota,
@naznachenie, @koleya, @speed)
              ELSE PRINT 'Скорость не может быть отрицательной'
            ELSE PRINT 'Колея состава не соответствует требованиям метро'
          ELSE PRINT 'Состав слишком высокий'
        ELSE PRINT 'Состав слишком широкий'
      ELSE PRINT 'Состав слишком длинный'
    ELSE PRINT 'Введите корректные значения'
  ELSE PRINT 'Тип с таким кодом уже существует!'
END
GO
```

```
CREATE PROC edit_typsostav (@typkod int, @name nvarchar(20), @kolmest int,
@dlina float, @shirina float, @visota float, @koleya int, @naznachenie nvarchar(20),
@speed int)
AS
BEGIN
  IF EXISTS (SELECT typkod FROM typsostav WHERE typkod=@typkod)
    IF (@dlina>0 and @shirina>0 and @visota>0 and @koleya>0 and @kolmest>0)
      IF @dlina < 200
        IF @shirina < 3000
          IF @visota < 5000
            IF @koleya=1520
              IF @speed>0
                UPDATE typsostav
                  SET name=@name, kolmest=@kolmest, dlina=@dlina, shirina=@shirina,
visota=@visota, koleya=@koleya, naznachenie=@naznachenie, speed=@speed
                  WHERE typkod=@typkod
              ELSE PRINT 'Скорость не может быть отрицательной'
            ELSE PRINT 'Колея состава не соответствует требованиям метро'
          ELSE PRINT 'Состав слишком высокий'
        ELSE PRINT 'Состав слишком широкий'
      ELSE PRINT 'Состав слишком длинный'
    ELSE PRINT 'Введите корректные значения'
  ELSE PRINT 'Типа с таким кодом не существует!'
END
GO

CREATE PROC del_typsostav (@typkod int)
AS
BEGIN
  IF EXISTS (SELECT typkod FROM typsostav WHERE typkod=@typkod)
    IF NOT EXISTS (SELECT serial FROM raspisanie WHERE serial=(SELECT TOP 1
serial FROM psostav WHERE typkod=@typkod))
```

```
    DELETE typsostav
      WHERE typkod=@typkod
   ELSE PRINT 'Удаление невозможно, так как составы данного типа
присутствуют в расписании!'
  ELSE PRINT 'Такого типа составов не существует'
END
GO
```

Создание функций пользователя

```
CREATE FUNCTION kol_stanciy (@vetka nvarchar(20))
RETURNS INT AS
BEGIN
  DECLARE @kol int, @stanciya int
  SET @kol=1
  SELECT @stanciya = start_stanciya FROM vetka WHERE name=@vetka
  WHILE (SELECT kod_vpered FROM stanciya WHERE kod=@stanciya) IS NOT
NULL
  BEGIN
    SET @kol = @kol + 1
    SET @stanciya=(SELECT kod_vpered FROM stanciya WHERE kod=@stanciya)
  END
  RETURN @kol
END
GO

CREATE FUNCTION adtime (@typ int, @stan int, @naprav int)
RETURNS INT AS
BEGIN
  DECLARE @s int
  IF (@naprav=0)
    SELECT @s=rast_vpered FROM stanciya WHERE kod=@stan
```

```
  ELSE
    SELECT @s=rast_nazad FROM stanciya WHERE kod=@stan
  DECLARE @u int
  SELECT @u=speed FROM typsostav WHERE typkod=@typ
  RETURN FLOOR(@s/@u)
END
GO

CREATE FUNCTION dlina_vetka (@vetka varchar(20))
RETURNS INT AS
BEGIN
  DECLARE @dlina int, @stanciya int
  SET @dlina=0
  SELECT @stanciya = start_stanciya FROM vetka WHERE name=@vetka
  WHILE (SELECT kod_vpered FROM stanciya WHERE kod=@stanciya) IS NOT
NULL
  BEGIN
    SET @dlina = @dlina + (SELECT rast_vpered FROM stanciya WHERE
kod=@stanciya)
    SET @stanciya=(SELECT kod_vpered FROM stanciya WHERE kod=@stanciya)
  END
  RETURN @dlina
END
GO

CREATE FUNCTION dlina (@stanciya int, @finish int, @p int, @p2 int)
RETURNS INT AS
BEGIN
  DECLARE @next_stanciya int, @rastoyanie int
  SET @rastoyanie=0;
  WHILE 1=1
  BEGIN
```

```
   IF (@stanciya=@finish and @p=@p2) BREAK
   IF (@p=0)
   BEGIN
    SELECT @next_stanciya=kod_vpered FROM stanciya WHERE kod=@stanciya
    IF (@next_stanciya is NULL)
    BEGIN
     SELECT @next_stanciya=kod_nazad FROM stanciya WHERE kod=@stanciya
     SET @p=1
     IF (@stanciya=@finish and @p=@p2) BREAK
     SET @rastoyanie=@rastoyanie+(SELECT rast_nazad FROM stanciya WHERE
kod=@stanciya)
    END
    ELSE SET @rastoyanie=@rastoyanie+(SELECT rast_vpered FROM stanciya
WHERE kod=@stanciya)
   END
   ELSE
   BEGIN
    SELECT @next_stanciya=kod_nazad FROM stanciya WHERE kod=@stanciya
    IF (@next_stanciya is NULL)
    BEGIN
     SELECT @next_stanciya=kod_vpered FROM stanciya WHERE kod=@stanciya
     SET @p=0
     IF (@stanciya=@finish and @p=@p2) BREAK
     SET @rastoyanie=@rastoyanie+(SELECT rast_vpered FROM stanciya WHERE
kod=@stanciya)
    END
    ELSE SET @rastoyanie=@rastoyanie+(SELECT rast_nazad FROM stanciya
WHERE kod=@stanciya)
   END
   SET @stanciya=@next_stanciya
  END
  RETURN @rastoyanie
```

```
END
GO

CREATE FUNCTION paspotok_vetki (@vetka nvarchar(20))
RETURNS INT AS
BEGIN
  DECLARE @paspotok int
  SET @paspotok=0;
  DECLARE @stanciya int
  SELECT @stanciya=start_stanciya FROM vetka WHERE name=@vetka
  WHILE @stanciya IS NOT NULL
  BEGIN
    IF (SELECT stat FROM stanciya WHERE kod=@stanciya) = 1
      SET @paspotok=@paspotok+(SELECT paspotok FROM stanciya WHERE
kod=@stanciya)
    SET @stanciya=(SELECT kod_vpered FROM stanciya WHERE kod=@stanciya)
  END
  RETURN @paspotok
END
GO

CREATE FUNCTION start_interval(@vetka nvarchar(20), @typ int, @kol_sostav int)
RETURNS INT AS
BEGIN
  DECLARE @speed int, @interval int, @s int, @f int
  SELECT @s=start_stanciya, @f=finish_stanciya FROM vetka WHERE
name=@vetka
  SELECT @speed=speed FROM typsostav WHERE typkod=@typ
  SET @interval = FLOOR((dbo.dlina_vetka(@vetka)*2)/@speed)
+15*dbo.kol_stanciy(@vetka)*2
  RETURN CEILING(@interval/@kol_sostav)
END
```

```
GO

CREATE FUNCTION maxi_sostav(@vetka nvarchar(20), @typ int)
RETURNS INT AS
BEGIN
  DECLARE @speed int, @interval int, @kol_sostav int, @s int, @f int
  SELECT @s=start_stanciya, @f=finish_stanciya FROM vetka WHERE
name=@vetka
  SELECT @speed=speed FROM typsostav WHERE typkod=@typ
  SET @interval = FLOOR((dbo.dlina_vetka(@vetka)*2)/@speed)
+15*dbo.kol_stanciy(@vetka)*2
  RETURN CEILING(@interval/90)
END
GO

CREATE FUNCTION proverka_sostav (@serial int)
RETURNS INT AS
BEGIN
  DECLARE @kod int
  SET @kod=0;
  IF (SELECT probeg FROM psostav WHERE serial=@serial) > (SELECT krit FROM
spravka WHERE kod=2)
    SET @kod=2;
  ELSE IF (SELECT iznos_motor FROM psostav WHERE serial=@serial) > (SELECT
krit FROM spravka WHERE kod=3)
      SET @kod=3;
    ELSE IF (SELECT iznos_tk FROM psostav WHERE serial=@serial) > (SELECT
krit FROM spravka WHERE kod=1)
      SET @kod=1;
  RETURN @kod
END
GO
```

```
CREATE FUNCTION sled_sostav (@ser int, @time time)
RETURNS INT AS
BEGIN
  DECLARE @vetka nvarchar(20), @nap int, @stanciya int
  SELECT  @stanciya=start_kod, @vetka=vetka, @nap=naprav FROM raspisanie
WHERE serial=@ser and start_vremya<DATEADD(SECOND,15,@time) and
finish_vremya>DATEADD(SECOND,-15,@time)
  DECLARE @temp int;
  SELECT TOP 1 @temp=serial
       FROM raspisanie
       WHERE start_vremya<DATEADD(SECOND,15,@time) and
finish_vremya>DATEADD(SECOND,-15,@time) and serial!=@ser
       ORDER BY dbo.dlina(@stanciya,start_kod,@nap,naprav) ASC
  RETURN @temp
END
GO
```

Создание триггеров

```
CREATE TRIGGER t_add_stanciya
ON stanciya INSTEAD OF INSERT
AS
DECLARE @kod int, @name varchar(20), @paspotok int, @kod_nazad int,
@rast_vpered int, @rast_nazad int, @stat int, @start_time time, @finish_time time
SELECT @kod = kod, @name=name, @paspotok=paspotok, @kod_nazad =
kod_nazad, @rast_vpered = rast_vpered, @rast_nazad = rast_nazad, @stat=stat,
@start_time=start_time, @finish_time=finish_time
  FROM inserted
IF (@kod_nazad IS NULL or @kod_nazad=0)
  INSERT INTO stanciya
```

```
   VALUES (@kod, @name, @paspotok, NULL, NULL, NULL, NULL, @stat,
@start_time, @finish_time)
ELSE
BEGIN
  DECLARE @s_kod int, @s_kod_vpered int, @s_kod_nazad int, @s_rast_vpered
int, @s_rast_nazad int
  DECLARE @CURSOR CURSOR
  SET @CURSOR  = CURSOR SCROLL
  FOR
  SELECT kod, kod_vpered, kod_nazad, rast_vpered, rast_nazad
    FROM stanciya
  OPEN @CURSOR
  FETCH NEXT FROM @CURSOR INTO @s_kod, @s_kod_vpered, @s_kod_nazad,
@s_rast_vpered, @s_rast_nazad
  WHILE @@FETCH_STATUS = 0
  BEGIN
    IF (@s_kod=@kod_nazad)
    BEGIN
      IF @rast_vpered IS NOT NULL
      BEGIN
        DECLARE @kod_vpered int
        SELECT @kod_vpered = kod_vpered
          FROM stanciya
          WHERE kod=@s_kod
        UPDATE stanciya SET kod_nazad = @kod WHERE kod = @kod_vpered
        UPDATE stanciya SET rast_nazad = @rast_vpered WHERE kod =
@kod_vpered
      END
      UPDATE stanciya SET kod_vpered = @kod WHERE kod=@s_kod
      UPDATE stanciya SET rast_vpered = @rast_nazad WHERE kod=@s_kod
      INSERT INTO stanciya
```

```
      VALUES (@kod, @name, @paspotok, @kod_vpered, @rast_vpered,
@kod_nazad, @rast_nazad, @stat, @start_time, @finish_time)
      BREAK
   END
   FETCH NEXT FROM @CURSOR INTO @s_kod, @s_kod_vpered,
@s_kod_nazad, @s_rast_vpered, @s_rast_nazad
 END
 CLOSE @CURSOR
END
GO

CREATE TRIGGER t_del_typsostav
ON typsostav INSTEAD OF DELETE
AS
DECLARE @typkod INT
SELECT @typkod=typkod
 FROM deleted
UPDATE mashinist
 SET serial=NULL
 WHERE serial IN (SELECT serial FROM psostav WHERE typkod=@typkod)
DELETE psostav
 WHERE typkod=@typkod
DELETE typsostav
 WHERE typkod=@typkod
GO

CREATE TRIGGER t_del_stanciya
ON stanciya INSTEAD OF DELETE
AS
DECLARE @kod int, @d_kod_vpered int, @d_kod_nazad int, @d_rast_vpered int,
@d_rast_nazad int
```

```
SELECT @kod=kod, @d_kod_nazad=kod_nazad, @d_kod_vpered=kod_vpered,
@d_rast_vpered=rast_vpered, @d_rast_nazad=rast_nazad FROM deleted
DECLARE @s_kod int, @s_kod_vpered int, @s_kod_nazad int, @s_rast_vpered int,
@s_rast_nazad int
DECLARE @CURSOR CURSOR
SET @CURSOR  = CURSOR SCROLL
FOR
SELECT kod, kod_vpered, kod_nazad, rast_vpered, rast_nazad
  FROM stanciya
OPEN @CURSOR
FETCH NEXT FROM @CURSOR INTO @s_kod, @s_kod_vpered, @s_kod_nazad,
@s_rast_vpered, @s_rast_nazad
WHILE @@FETCH_STATUS = 0
BEGIN
  IF @s_kod=@d_kod_nazad
    UPDATE stanciya
      SET kod_vpered=@d_kod_vpered, rast_vpered=rast_vpered+@d_rast_vpered
      WHERE kod=@s_kod
  IF @s_kod=@d_kod_vpered
    UPDATE stanciya
      SET kod_nazad=@d_kod_nazad, rast_nazad=rast_nazad+@d_rast_nazad
      WHERE kod=@s_kod
  FETCH NEXT FROM @CURSOR INTO @s_kod, @s_kod_vpered, @s_kod_nazad,
@s_rast_vpered, @s_rast_nazad
END
  DELETE stanciya
    WHERE kod=@kod
CLOSE @CURSOR
GO
```

Приложение 2. Сценарий заполнения БД

```
USE Metro
GO

INSERT INTO stanciya (kod, name, paspotok, kod_vpered, rast_vpered,
kod_nazad, rast_nazad, stat)
VALUES
('1', 'Парк Культуры', '49', '2', '1000', NULL, NULL, 1),
('2', 'Кировская', '69', '3', '750', '1', '1000', 1),
('3', 'Комсомольская', '89', '4', '1000', '2', '750', 1),
('4', 'Автозаводская', '98', '5', '700', '3', '1000', 1),
('5', 'Пролетарская', '98', '6', '900', '4', '700', 1),
('6', 'Двигатель Революции', '139', '7', '1100', '5', '900', 1),
('7', 'Заречная', '218', '8', '1600', '6', '1100', 1),
('8', 'Ленинская', '219', '9', '1550', '7', '1600', 1),
('9', 'Чкаловская', '123', '10', '1000', '8', '1550', 1),
('10', 'Московская', '84', '11', '3000', '9', '1000', 1),
('11', 'Горьковская', '47', NULL, NULL, '10', '3000', 1),
('12', 'Буревестник', '59', '13', '1150', NULL, NULL, 1),
('13', 'Бурнаковская', '95', '14', '1150', '12', '1150', 1),
('14', 'Канавинская', '149', '15', '1200', '13', '1150', 1),
('15', 'Московская', '200', '16', '2500', '14', '1200', 1),
('16', 'Стрелка', '117', '17', '2300', '15', '2500', 1),
('17', 'Волга', '68', NULL, NULL, '16', '2300', 1)
GO


INSERT INTO vetka
VALUES
('Красная', 1, 11, 1, 3, 5),
```

('Синяя', 12, 17, 2, 4, 6)
GO

INSERT INTO dispetcher
VALUES
(123456, 'Александр Акимов', '89102348768', 'Старший Диспетчер',
HASHBYTES('MD5','123456')),
(748120, 'Евгений Кулик', '89109876543', 'Диспетчер',
HASHBYTES('MD5','748120')),
(321295, 'Анастасия Азимова', '89103219876', 'Диспетчер',
HASHBYTES('MD5','321295')),
(398123, 'Евгений Кулик', '89158374921', 'Диспетчер',
HASHBYTES('MD5','398123')),
(129348, 'Юлия Нагибина', '83927430134', 'Диспетчер',
HASHBYTES('MD5','129348'))
GO

INSERT INTO typsostav
VALUES
('1', '81-767 «Москва»', '330', '160', '2686', '3680', 'Пасажирский', '1520', '11'),
('2', '81-717', '330', '192', '2670', '3650', 'Пасажирский', '1520', '13'),
('5', 'E81-717', '330', '192', '2670', '3650', 'Пасажирский Автоуправляемый',
'1520', '13'),
('3', 'CMM-2', '0', '30', '2590', '3690', 'Снегоуборочный', '1520', '7'),
('4', 'RYT-4', '0', '30', '2700', '3700', 'Промывка Тонелей', '1520', '7')
GO

INSERT INTO psostav
VALUES
('1', '1', '1', '20432', '20', '4'),
('2', '1', '1', '12342', '15', '5'),

('3', '1', '1', '20394', '18', '2'),
('4', '1', '1', '13321', '17', '3'),
('5', '1', '1', '12493', '18', '4'),
('6', '1', '1', '12943', '18', '5'),
('7', '1', '1', '18242', '20', '2'),
('8', '1', '1', '28432', '22', '2'),
('9', '1', '1', '21827', '15', '3'),
('10', '1', '1', '18212', '12', '3'),
('11', '2', '1', '9123', '5', '5'),
('12', '2', '1', '8323', '2', '4'),
('13', '2', '1', '10212', '3', '3'),
('14', '2', '1', '7293', '4', '4'),
('15', '2', '2', '3928', '5', 37),
('16', '2', '2', '9432', '9', '4'),
('17', '2', '2', '3023', '12', '2'),
('18', '2', '2', '3943', '2', '4'),
('19', '2', '2', '1293', '7', '5'),
('20', '2', '2', '9432', '8', '6'),
('21', '3', '2', '5034', '10', '3'),
('22', '3', '2', '7054', '15', '4'),
('23', '4', '2', '15033', '20', '5'),
('24', '4', '2', '13495', '22', '2')
GO

INSERT INTO mashinist
VALUES
(340432, '1', 'Павел Дуров', '89495571599', '7', 'Третья'),
(385312, '2', 'Дмитрий Медведев', '', '4', 'Первая'),
(586341, '3', 'Михаил Галустян', '86128324123', '8', 'Вторая'),
(963453, '4', 'Макс Тарасенко', '89320432123', '1', 'Первая'),
(3582395823, '5', 'Егор Крид', '89102938473', '2', 'Первая'),
(4599683475, '6', 'Павел Воля', '89159432384', '8', 'Вторая'),

(2395834523, '7', 'Максим Голополосов', '83928438899', '15', 'Третья'),
(2387574234, '8', 'Роман Фильченков', '', '1', 'Первая'),
(4958873644, '9', 'Андрей Рогозов', '', '3', 'Первая'),
(4495948242, '10', 'Рамзан Кадыров', '8394838412', '3', 'Первая'),
(3988574444, '11', 'Алексей Долматов', '83743745646', '4', 'Первая'),
(3384754332, '12', 'Дима Билан', '', '9', 'Третья'),
(2340542485, '13', 'Александр Шепс', '83748374212', '5', 'Первая'),
(3843743244, '14', 'Николай Соболев', '89102837463', '8', 'Вторая'),
(3847738495, '15', 'Вася Вакуленко', '87364521243', '3', 'Первая'),
(4838475554, '16', 'Владимир Жириновский', '', '25', 'Первая'),
(2283746372, '17', 'Юрий Хованский', '', '15', 'Третья'),
(3948577483, '18', 'Макс Брандт', '', '5', 'Первая'),
(2398327475, '19', 'Даниил Добродушный', '89057349394', '4', 'Первая'),
(4938472384, '20', 'Николай Наумов', '89056743432', '3', 'Первая'),
(4039857422, '21', 'Алексей Навальный', '89204930794', '9', 'Вторая'),
(9584727343, '22', 'Костя Павлов', '89208956350', '14', 'Третья'),
(2944723954, '23', 'Рома Жёлудь', '89027853099', '25', 'Третья'),
(2563323954, NULL, 'Сергей Собянин', '89084105484', '34', 'Третья')
GO

INSERT INTO spravka
VALUES
(1, 'Колодки', '35', 10, 15),
(2, 'Пробег', '200000', 0, 23),
(3, 'Мотор', '25', 25, 20)
GO

INSERT INTO remont (typik)
VALUES
(1), (2), (3), (4), (5)
GO