**Автономная некоммерческая образовательная**

**организация высшего образования**

**«Сахалинский гуманитарно-технологический институт»**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ФАКУЛЬТЕТ ЭКОНОМИКИ И ИНФОРМАТИКИ**

**Кафедра компьютерные технологии и системы**

Никонов Василий Сергеевич

**РАЗРАБОТКА БАЗЫ ДАННЫХ «РАСПИСАНИЕ ДВИЖЕНИЯ АВТОБУСОВ»**

**КУРСОВАЯ РАБОТА**

по дисциплине «Базы данных»

Студент гр. ИВТ-31 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Руководитель ст. преподаватель

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ П.Ю. Соболев

Регистрационный № \_\_\_\_\_\_\_ Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись И.О.Фамилия

\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2018 г.  подпись И.О.Фамилия

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2018 г.

г. Южно-Сахалинск

2018 г.

**«Сахалинский гуманитарно-технологический институт»**

*Кафедра «Компьютерные технологии и системы»*

УТВЕРЖДЕНО

Зав. кафедрой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Мазур И.К.

**З А Д А Н И Е**

на выполнение курсовой работы студенту: *Никонову Василию Сергеевичу*

1. Тема работы: *«Разработка базы данных «расписание движения автобусов»*

2. Срок сдачи студентом законченного проекта: *29 мая 2018*

3. Содержание пояснительной записки (перечень подлежащих к разработке вопросов):

*Введение*

*1. Анализ и описание предметной области*

*2. Анализ и проектирование данных, разработка хранимых процедур и представлений*

*3. Разработка программного обеспечения*

*Заключение е*

Задание принял к исполнению: 30/01/2018

(дата, подпись)

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

***ПРИМЕЧАНИЕ: Студент предупрежден о том, что при несвоевременном представлении работы руководителю студент считается не выполнившим график учебного процесса и не допускается к защите курсовой работы.***

**ОГЛАВЛЕНИЕ**

[1 Постановка задачи 4](#_Toc311632381)

[2 Анализ предметной области 5](#_Toc311632382)

[3 Проектирование базы данных 11](#_Toc311632383)

[4 Реализация базы данных в формате MS SQL Server 29](#_Toc311632384)

[5 Хранимые процедуры для вставки данных 36](#_Toc311632385)

[6 Проектирование программного обеспечения 68](#_Toc311632386)

[7 Руководство пользователя 72](#_Toc311632387)

[Список литературы 77](#_Toc311632388)

[Приложение 78](#_Toc311632389)

# 1 Постановка задачи

Предметная область, подлежащая изучению – «транспортная компания». Цель этой предметной области – отслеживание перемещения автобусов по городским маршрутам. В отслеживаемую информацию входят данные о маршрутах и их остановках, данные об автобусе, и работников, работающих на них.

***Целью курсовой работы является*** создание информационно-программной системы для решения следующих задач:

1. автоматизации отслеживания перемещения автобусов:
   1. пункт прибытия,
   2. время прибытия,
   3. время отправления,
   4. маршрут перемещения;
2. ведение журнала перемещения автобуса по город,
3. отслеживания работников предприятия и техники на которой они работают,
4. поиск и выборка данных по различным критериям,
5. формирование отчетов.

К разрабатываемой системе предъявлены следующие требования:

1. наличие централизованного хранилища данных для всех объектов автоматизированного учета,
2. ведение операций информационного учета с компьютеров локальной сети предприятия,
3. создание унифицированного доступа к данным с использованием специализированного программного интерфейса пользователя, который бы отражал специфику этой предметной области.

Для решения поставленных задач планируется выполнение следующих действий:

1. проектирование данных для отражения специфики данной предметной области,
2. разработка и создание базы данных для хранения информации на едином сервере данных под управлением MS SQL Server ,
3. разработка компьютерной программы для выполнения базовых операций над данными предметной области, которая выполняет следующие функции:
   1. добавление информации об основных объектах учёта,
   2. редактирование информации по основным объектам учёта,
   3. поиск данных по основным объектам учёта,
   4. получение итоговой информации по отчетным характеристикам системы.

**2 Анализ предметной области**

В ходе проведённого анализа предметной области было выявлено, что транспортная компания и отслеживание перемещений автобусов - обширная область. Её основные направления — отслеживание маршрутов, автобусов и персонала. Было выявлено несколько сущностей, составляющих приблизительный глоссарий предметной области.

Таблица 1 – Глоссарий предметной области

|  |  |
| --- | --- |
| **Ключевое слово** | **Вид** |
| Пассажир | бизнес-актёр |
| Кондуктор | бизнес-актрер |
| Произвести оплату | бизнес-прецедент |
| Прием оплаты | бизнес-прецедент |
| Обслуживание автобусов | бизнес-прецедент |
| Водит автобус | бизнес-прецедент |
| Выплачивается зарплата | бизнес-прецедент |
| Техническое обслуживание | бизнес-прецедент |
| Заполнить путевой лист | бизнес-прецедент |
| Прием путевого листа листа | бизнес-прецедент |
| Оплата | бизнес-сущность |
| Автобус | Бизнес-сущность |
| Маршрут | Бизнес-сущность |
| Остановка | Бизнес-сущность |
| Водитель | Бизнес-сущность |
| Зарплата | Бизнес-сущность |
| СТО | Бизнес-сущность |
| Путевой лист | Бизнес-сущность |
| Депо | Бизнес-сущность |
| Имеет маршрут | Категория бизнес-сущности |
| Содержит остановку | Категория бизнес-сущности |

Целью существования системы отслеживания автобусов является повышение качества предоставляемых услуг автобусной компании. Показателем успеха в достижении этой цели является увеличение пассажиропотока. Так же успех предприятия зависит от точности расписания маршрутов. Оно должно быть актуальным и обновляться в зависимости от загруженности дорог города.

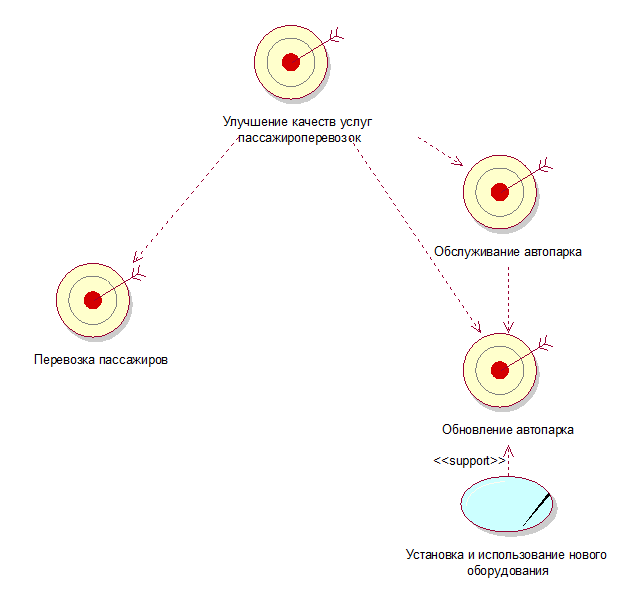


Рисунок 1 – UML диаграмма бизнес-целей предметной области

Касательно деятельности предметной области, можно рассмотреть три основных направления: перевозка пассажиров, обработка информации о перемещении автопарка и управление автопарком предприятия. Кроме этого, существует ещё несколько направлений, в курсовой работе не рассмотренных: хозяйственная деятельность, не связанная с прямой задачей системы, рекламные кампании, и другие.

Деятельность перевозчика (см. рисунок 2) основывается на взаимосвязанных усилиях всех сотрудников: билетёры или кондукторы принимают оплату за проезд, водители останавливают транспортное средство на стандартных остановках маршрута, а также могут остановить на остановках по требованию. Пассажир заплатив за билет, может сойти на любой остановке маршрута автобуса, в том числе и по требованию.

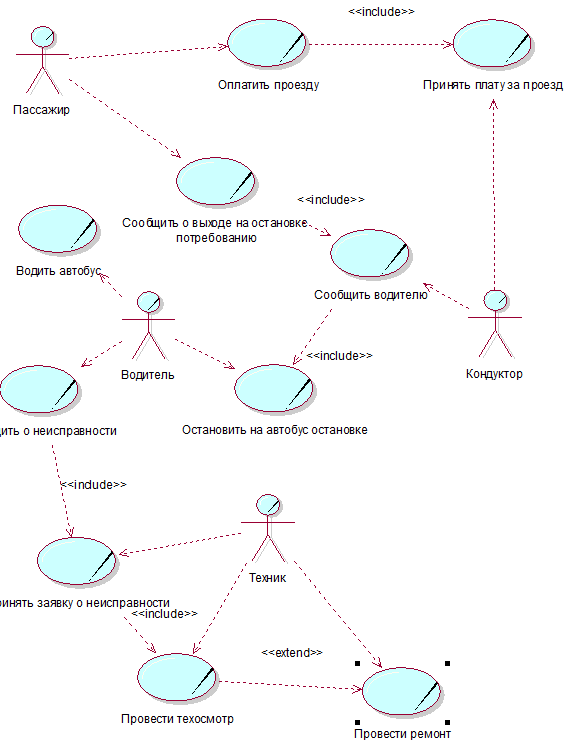


Рисунок 2 – UML диаграмма бизнес-прецедентов деятельности перевозчика

Основной целью создаваемой базы данных является сбор сведений о перемещения автобусов по маршруту. Данные для обработки собираются в автоматическом режиме с помощью специализированного оборудования.

**3 Проектирование базы данных**

Создаваемая база данных предназначена для автоматизации учётной деятельности перевозчика, хранения и обработки всей информации об автопарке, маршрутов, и передвижениях автобусов по ним, а также формирования отчётов.

Пользователями базы данных могут быть те сотрудники компании, или автоматические системы выдачи информации, такие как электронные табло, выдающие расписания автобусов.

Помимо базы данных, требуется создание приложения, обеспечивающего пользовательский доступ к базе данных, предоставляющий добавление, изменение и удаление, сопровождающихся соответствующими отчётами. Возможность гибкого поиска по множеству категорий значительно упростит работу с данными. Для специалистов компании будет представлена специальная форма, содержащие необходимые для иследования данные.

Таблица 2 – описание сущностей предметной области

|  |  |  |  |
| --- | --- | --- | --- |
| ***Название***  ***сущности*** | ***Описание*** | ***Вид*** | ***Зависимые***  ***сущности*** |
| Депо | Гаражи организации | Стержневая | Автобус |
| Автобус | Транспорт организации использоуемый для осуществления бизнес деятельности | Стержневая | Рейс |
| Тип автобуса | Тип транспортного средства и его характеристика | Характеристическая (справочник) | Автобус |
| Рейс | Путь автобуса по определенному маршруту | стержневая | Работники\_на\_рейсе, Перемещения\_по\_маршруту |
| Работники\_на\_рейсе | Работник прикреплянный к рейсу, связывает Работника и Рейс | ассоциативная | нет |
| Работник | Работник организации | стержневая | Работник\_на\_рейсе |
| Маршрут | Маршрут автобуса по городу | стрежневая | Рейс, Остановка\_маршрута |

Окончание таблицы 2

|  |  |  |  |
| --- | --- | --- | --- |
| ***Название***  ***сущности*** | ***писание*** | ***Вид*** | ***Зависимые***  ***сущности*** |
| Остановка маршрута | Остановка на маршруте следования, связывает маршрут и остановку | ассоциативная | Перемещения\_по\_маршруту |
| Остановка | Остановка для посадки и высадки пассажиров | стержневая | Остановка\_Маршрута |
| Перемещения по маршруту | Информация о перемещениях автобуса по маршруту, связывает Остановку маршрута и рейс | ассоциативная | Нет |
| Должности | Информация о должностях сотрудников компании | стержневая | Сотрудник |

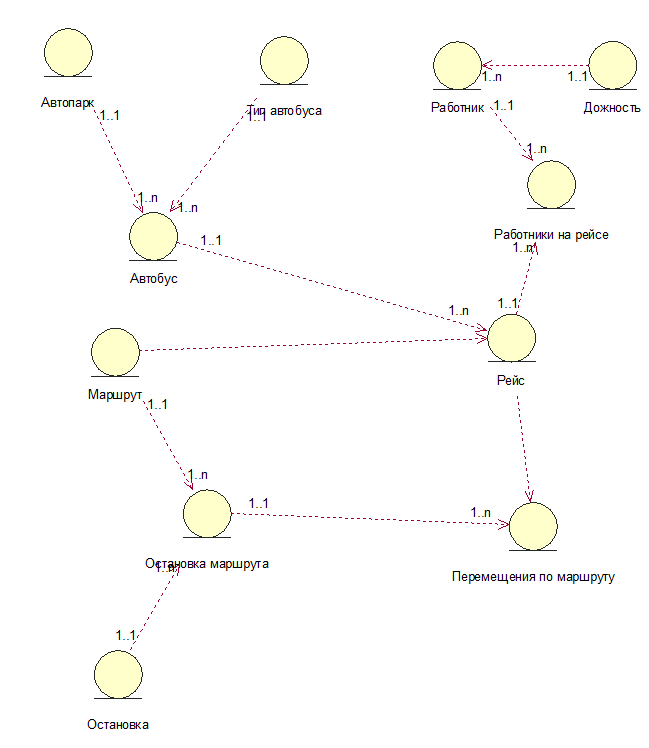


Рисунок 3 – диаграмма классов предметной области

Рассмотрим диаграмму классов предметной области (см. рисунок 3), которая отображает информационно-логическую структуру изучаемой предметной области с точки зрения обработки и хранения информации о её бизнес-процессах.

Ранее, на этапе анализа, в составе предметной области была вывялена сущность «*депо*». В соответствии с принципом семантического моделирования, сопоставим ей, в информационно-логической модели, отношение «*депо*», схема которого показана далее в таблице 3.

Таблица 3 – Схема отношения «депо»

|  |  |  |
| --- | --- | --- |
| **Атрибут** | **Смысл** | **Тип** |
| Номер Депо | Номер гаража на территории предприятия (PK) | числовой целый |
| Вместимость | Вместимость гаража | числовой целый |
| Количество техников | Количество техников работающих в гараже | числовой целый |

Схема удовлетворяет 1НФ, 2НФ, 3НФ, так как все атрибуты атомарны, все неключевые атрибуты зависят от первичного ключа, и нет функциональных зависимостей между неключевыми атрибутами.

Таблица 4 – Схема отношения «Автобус»

|  |  |  |
| --- | --- | --- |
| **Атрибут** | **Смысл** | **Тип** |
| Гос\_номер | Государственный номер машины(PK) | Строка |
| Гаражный номер | Гаражный номер машины предприятия | Строка |
| Тип\_автобуса | Тип автобуса | Числовой целый |
| Номер депо | Номер депо к которому прикреплен автобус | Числовой целый |

1. Среди атрибутов нет составных, таким образом, создано корректное отношение в 1НФ,
2. Отношение «*Автобус*» находится в 2НФ, так как все неключевые атрибуты данного отношения полностью зависят от первичного ключа,
3. Между не ключевыми атрибутами отношения не существует функциональных зависимостей. Учитывая пункт 2, можно утверждать, что данное отношение находится в 3НФ, что и требуется.

Таблица 5 – Схема отношения «Тип автобуса»

|  |  |  |
| --- | --- | --- |
| **Атрибут** | **Смысл** | **Тип** |
| Тип автобуса | Название типа автобуса (PK) | Строка |
| Общая вместимость | Вместимость автобуса данного типа | числовой целый |
| Сидячих мест | Места для посадки пассажиров в данном типе автобусов | числовой целый |

1. Среди атрибутов нет составных, таким образом, создано корректное отношение в 1НФ,
2. Отношение «*тип автобуса*» находится в 2НФ, так как все не ключевые атрибуты данного отношения полностью зависят от первичного ключа,
3. Между не ключевыми атрибутами отношения не существует функциональных зависимостей. Учитывая пункт 2, можно утверждать, что данное отношение находится в 3НФ, что и требуется.

Таблица 6 – Схема отношения «Рейс»

|  |  |  |
| --- | --- | --- |
| **Атрибут** | **Смысл** | **Тип** |
| Номер путевого листа | Номер путевого листа рейса автобуса (PK) | Строка |
| Гос номер | Государственный номер автобуса отправленного в рейс (значение ключевого атрибута отношения «Автобус») (FK) | Строка |
| Номер маршрута | Номер маршрута следования автобуса (значение ключевого атрибута отношения «Маршрут») (FK) | Строка |
| Дата Рейса | Дата совершения рейса | Дата |

Схема содержит два внешних ключа: первичные ключи из схем «Автобус» и «Маршрут».

1. Среди атрибутов нет составных, таким образом, создано корректное отношение в 1НФ,
2. Отношение «*рейс*» находится в 2НФ, так как все не ключевые атрибуты данного отношения полностью зависят от первичного ключа,
3. Между не ключевыми атрибутами отношения не существует функциональных зависимостей. Учитывая пункт 2, можно утверждать, что данное отношение находится в 3НФ, что и требуется.

Таблица 7 – Схема отношения «Работники на рейсе»

|  |  |  |
| --- | --- | --- |
| **Атрибут** | **Смысл** | **Тип** |
| id\_смены | Используется для однозначной идентификации Работника на рейсе в информационной системе (PK) | Числовой целый, счётчик |
| Номер путевого листа | Номер путевого листа рейса (значение ключевого атрибута отношения «Рейс») (FK) | Строка |
| ИНН\_работника | Индивидуальный номер налогоплательщика (значение ключевого атрибута отношения «Работник») (FK) | Числовой целый |

Для идентификации работника на рейсе в отношение введен атрибут «id\_смены». Этот атрибут является первичным ключом данного отношения, так как обладает свойством неизбыточности и позволяет однозначно идентифицировать отдельного автора по его целочисленному номеру. Схема содержит внешние ключи: ссылку на первичный ключ из схемы «Рейс», а также ссылку на первичный ключ из схемы «Работник».

1. Среди атрибутов нет составных, таким образом, создано корректное отношение в 1НФ,
2. Отношение «*работник на рейсе*» находится в 2НФ, так как все неключевые атрибуты данного отношения полностью зависят от первичного ключа,
3. Между неключевыми атрибутами отношения не существует функциональных зависимостей. Учитывая пункт 2, можно утверждать, что данное отношение находится в 3НФ, что и требуется.

Таблица 8 – Схема отношения «Работник»

|  |  |  |
| --- | --- | --- |
| **Атрибут** | **Смысл** | **Тип** |
| ИНН работника | Индивидуальный Номер Налогоплательщика работника организации | Числовой целый |
| Фамилия | Фамилия работника | Строка |
| Имя | Имя работника | Строка |
| Отчество | Отчество работника | Строка |
| Название должности | Должность работника предприятия (значение ключевого атрибута отношения «Должность») (FK) | Строка |
| Дата приема на работу | Дата приема на работу работника | Дата |
| Дата увольнения работника | Дата увольнения работника | Дата |
| фотография | фотография сотрудника | изображение |

1. Среди атрибутов нет составных, таким образом, создано корректное отношение в 1НФ,
2. Отношение «*Работник*» находится в 2НФ, так как все неключевые атрибуты данного отношения полностью зависят от первичного ключа,
3. Между неключевыми атрибутами отношения не существует функциональных зависимостей. Учитывая пункт 2, можно утверждать, что данное отношение находится в 3НФ, что и требуется.

Таблица 9 – Схема отношения «Маршрут»

|  |  |  |
| --- | --- | --- |
| **Атрибут** | **Смысл** | **Тип** |
| Номер маршрута | Номер маршрута следования автобусов (PK) | строка |
| Длина маршрута | Протяженность маршрута в км | числовой, вещественный |

Сущность, соответствующая данной схеме отношений, является стержневой. Первичным ключом номер маршрута.

1. Среди атрибутов нет составных, таким образом, создано корректное отношение в 1НФ,
2. Отношение «*маршрут*» находится в 2НФ, так как все неключевые атрибуты данного отношения полностью зависят от первичного ключа,
3. Между неключевыми атрибутами отношения не существует функциональных зависимостей. Учитывая пункт 2, можно утверждать, что данное отношение находится в 3НФ, что и требуется.

Таблица 10 – Схема отношения «остановка маршрута»

|  |  |  |
| --- | --- | --- |
| **Атрибут** | **Смысл** | **Тип** |
| Номер маршрута | Номер маршрута следования (PK) | Строка |
| id\_остановки | уникальный индификатор остановочного пункта (PK) | числовой целый, счетчик |

1. Среди атрибутов нет составных, таким образом, создано корректное отношение в 1НФ,
2. Отношение «*остановка маршрута*» находится в 2НФ, так как все неключевые атрибуты данного отношения полностью зависят от первичного ключа,
3. Между неключевыми атрибутами отношения не существует функциональных зависимостей. Учитывая пункт 2, можно утверждать, что данное отношение находится в 3НФ, что и требуется.

Таблица 11 – Схема отношения «остановка»

|  |  |  |
| --- | --- | --- |
| **Атрибут** | **Смысл** | **Тип** |
| id\_остановки | Уникальный индефикатор остановочного пункта | числовой, целый, счетчик |
| Название\_остановки | название остановочного пунктана маршруте | строка |

1. Среди атрибутов нет составных, таким образом, создано корректное отношение в 1НФ,
2. Отношение «*остановка*» находится в 2НФ, так как все неключевые атрибуты данного отношения полностью зависят от первичного ключа,
3. Между неключевыми атрибутами отношения не существует функциональных зависимостей. Учитывая пункт 2, можно утверждать, что данное отношение находится в 3НФ, что и требуется.

Таблица 12 – Схема отношения «перемещения по маршруту»

|  |  |  |
| --- | --- | --- |
| **Атрибут** | **Смысл** | **Тип** |
| ид\_перемещения\_по\_маршруту | индификационный номер для однозначного определения записи в системе (PK) | числовой целый, счетчик |
| ид\_остановки | индификационный номер для однозначного определения записи в системе | числовой целый |
| номер путевого листа | номер путевого листа по которому совершаеться рейс на транспорте организации | строка |
| время\_прибытия | время прибытия на остановку | время |
| время\_отбытия | время отбытия с остановки | время |

1. Среди атрибутов нет составных, таким образом, создано корректное отношение в 1НФ,
2. Отношение «*перемещение по маршруту*» находится в 2НФ, так как все неключевые атрибуты данного отношения полностью зависят от первичного ключа,
3. Между неключевыми атрибутами отношения не существует функциональных зависимостей. Учитывая пункт 2, можно утверждать, что данное отношение находится в 3НФ, что и требуется.

Таблица 13 – Схема отношения «должности»

|  |  |  |
| --- | --- | --- |
| **Атрибут** | **Смысл** | **Тип** |
| название\_должности | Название должности в огрганизации (PK) | строка |
| оклад | оклад сотрудника на должности | денежный |

1. Среди атрибутов нет составных, таким образом, создано корректное отношение в 1НФ,
2. Отношение «*должность*» находится в 2НФ, так как все неключевые атрибуты данного отношения полностью зависят от первичного ключа,
3. Между неключевыми атрибутами отношения не существует функциональных зависимостей. Учитывая пункт 2, можно утверждать, что данное отношение находится в 3НФ, что и требуется.

Итак, в изучаемой предметной области выявлены и формализованы данных выявлены многочисленные связи между сущностями. Рассмотрим каждую связь и установим тип каждой из них в соотвествии с правила реляционной теории. Рассмотрим связь «Тип\_автобуса-Автобус» (рисунок 4). Сущность «тип\_автобуса» является небольшим словарём типов источника для сущности «Автобус». В связи «Тип\_автобуса-Автобус», главной является сущность «Тип\_автобуса», а подчинённой является сущность «Автобус» (так как тип\_автобуса является атрибутом автобуса). Данная связь имеет тип «один ко многим», так как каждый автобус может иметь только один тип, а каждый тип может принадлежать нескольким автобусам. Для реализации данной связи в составе подчинённого отношения «*автобус*» необходимо ввести внешний ключ, который означает тип автобуса, который он имеет (таким атрибутом является «id\_типа\_автобуса» в отношении «*тип\_автобуса*»).

тип\_автобуса

автобус

пренадлежит

1

n

Рисунок 4 – смысл связи «тип\_автобуса-автобус»

Рассмотрим связь «Депо-Автобус» (рисунок 5). В связи «депо-автобус», главной является сущность «депо», а подчинённой является сущность «автобус» (так как номер депо является атрибутом автобуса). Данная связь имеет тип «один ко многим», так как каждый автобус может принадлежать только одному депо, а каждое депо может обладать несколькими автобусами. Для реализации данной связи в составе подчинённого отношения «*автобус*» необходимо ввести внешний ключ, который означает депо, к которой он принадлежит (таким атрибутом является «Номер\_депо» в отношении «депо»).

Депо

автобус

имеет

1

n

Рисунок 5 – смысл связи «Депо-автобус»

Рассмотрим связь «должности-работник» (рисунок 6). В связи «должности-работник», главной является сущность «должности», а подчинённой является сущность «работник» (так как должность является атрибутом работника). Данная связь имеет тип «один ко многим», так как каждый работник может иметь только одну должность, а одна должность может быть у разных рабочих. Для реализации данной связи в составе подчинённого отношения «*работник*» необходимо ввести внешний ключ, который означает должность (таким атрибутом является «id\_должности» в отношении «*должность*»).

должность

работник

характеризует

1

n

Рисунок 6 – смысл связи «должность-работник»

Рассмотрим связь «остановка-остановка\_маршрута» (рисунок 7). Сущность «остановка маршрута» является ассоциативной сущностью для сущностей «остановка» и «маршрут» для разбиения связи «многие ко многим». В связи «остановка-остановка\_маршрута», главной является сущность «остановка», а подчинённой является сущность «Остановка\_маршрута» (так как остановка является атрибутом остановки маршрута). Данная связь имеет тип «один ко многим», так как каждая остановка маршрута может принадлежать только одномой остановке, а каждая остановка может обладать несколькими остановками маршрута. Для реализации данной связи в составе подчинённого отношения «остановка\_маршрута» необходимо ввести внешний ключ, который означает остановку, к которой она принадлежит (таким атрибутом является «ид остановки» в отношении «Остановка»).

Остановка*авителямитилясущностью онат.содержит ссылки на схемы "чных ключей соответствующих схем.ти "ции, а для научного сотрудника,*

Остановка маршрута

характеризует

1

n

Рисунок 7 – смысл связи «остановка-остановка маршрута»

Рассмотрим связь «маршрут-остановка маршрута» (рисунок 8). Сущность «остановка маршрута» является ассоциативной сущностью для сущностей «маршрут» и «остановка» для разбиения связи «многие ко многим». В связи «маршрут-остановка маршрута», главной является сущность «маршрут», а подчинённой является сущность «остановкой маршрута» (так как маршрут является атрибутом остановки маршрута). Данная связь имеет тип «один ко многим», так один маршрут может содежать несколько остановок, а одна остановка маршрута может входить в несколько разных маршрутов. Для реализации данной связи в составе подчинённого отношения «остановка маршрута» необходимо ввести внешний ключ, который означает маршрут, которому он принадлежит (таким атрибутом является «номер маршрута» в отношении «маршрут»).

маршрут

Остановка маршрута

включает

1

n

Рисунок 8 – смысл связи «маршрут-остановка маршрута»

Рассмотрим связь «работник-работник на рейсе» (рисунок 9). Отношение «работник на рейсе» существует для разбиения связи многие ко многим. В связи «работник-работник на рейсе», главной является сущность «работник», а подчинённой является сущность «работник на рейсе» (так как работник является атрибутом работника на рейсе). Данная связь имеет тип «один ко многим», так как каждый работник может принадлежать только одному работнику на рейсе, а в работника на рейсе может входить несколько работников. Для реализации данной связи в составе подчинённого отношения «работник на рейсе» необходимо ввести внешний ключ, который означает работника (таким атрибутом является «ИНН работника» в отношении «работник»).

Работник

Работник на рейсе

характеризует

1

n

Рисунок 9 – смысл связи «Работник-работник на рейсе»

Рассмотрим связь «автобус-рейс» (рисунок 10). В связи «автобус-рейс», главной является сущность «автобус», а подчинённой является сущность «рейс» (так как стиль является атрибутом экспоната). Данная связь имеет тип «один ко многим», так как каждый экспонат может принадлежать только одному стилю, а каждый стиль может обладать несколькими экспонатами. Для реализации данной связи в составе подчинённого отношения «*экспонат*» необходимо ввести внешний ключ, который означает стиль, к которому он принадлежит (таким атрибутом является «название\_стиля» в отношении «стиль»).

автобус

Рейс

пренадлежит

1

n

Рисунок 10 – смысл связи «автобус-рейс»

Рассмотрим связь «остановка\_маршрута-перемещение\_по\_маршруту» (рисунок 11). В связи «остановка\_маршрута-перемещение\_по\_маршруту», главной является сущность «остановка», а подчинённой является сущность «перемещение по маршруту» (так как остановка является атрибутом перемещением по маршруту). Данная связь имеет тип «один ко многим», так как несколько перемещений по маршруту могут иметь одну остановку, а каждый остановка может быть включена в несколько перемещений по маршруту. Для реализации данной связи в составе подчинённого отношения «перемещения по маршруту» необходимо ввести внешний ключ, который означает остановку, к которому он относиться (таким атрибутом является «ид остановки» в отношении «остановка»).

остановка

Перемещения по маршруту

пренадлежит

1

n

Рисунок 11 – смысл связи «остановка-перемещения по маршруту»

Рассмотрим связь «рейс-работник на рейсе» (рисунок 12). В связи «рейс-работник на рейсе», главной является сущность «рейс», а подчинённой является сущность «работник на рейсе» (так как рейс является атрибутом работника на рейсе). Данная связь имеет тип «один ко многим», так как каждый работник на рейсе может принадлежать только одной рейсу, а каждый работник может входить в несколько рейсов. Для реализации данной связи в составе подчинённого отношения «*работник на рейсе*» необходимо ввести внешний ключ, который означает рейс, к которому он принадлежит (таким атрибутом является «номер путевого листа» в отношении «рейс»).

рейс

Работник на рейсе

характеризует

1

n

Рисунок 12 – смысл связи «рейс-работник на рейсе»

Рассмотрим связь «рейс-перемещение по маршруту» (рисунок 13). В связи «рейс-перемещение по маршруту», главной является сущность «рейс», а подчинённой является сущность «перемещение по маршруту» (так как рейс является атрибутом перемещения по маршруту). Данная связь имеет тип «один ко многим», так как каждое перемещение по маршуруту может состоять только из одного рейса, а каждый рейс может обладать несколькими перемещениями по рейсу. Для реализации данной связи в составе подчинённого отношения «*перемещение по рейсу*» необходимо ввести внешний ключ, который означает рейс, из в который он входит (таким атрибутом является «Номер путевого листа» в отношении «рейс»).

рейс

Перемещение по рейсу

входит

1

n

Рисунок 13 – смысл связи «рейс-перемещение по рейсу»

Рассмотрим связь «маршрут-рейс» (рисунок 14). В связи «маршрут-рейс», главной является сущность «маршрут», а подчинённой является сущность «рейс» (так как рейсявляется атрибутом маршрута). Данная связь имеет тип «один ко многим», так как каждый рейс может входить только в один маршрут, а каждый маршрут может порождать несколько рейсов. Для реализации данной связи в составе подчинённого отношения «*рейс*» необходимо ввести внешний ключ, который означает маршрут, в который входит рейс (таким атрибутом является «Номер маршрута» в отношении «маршрут»).

маршрут

рейс

порождает

1

n

Рисунок 14 – смысл связи «маршрут-рейс»

**4 Реализация базы данных в формате MS SQL Server**

На этом этапе следует реализовать созданную ранее структуру на сервере БД MS SQL Server в базе данных. Далее перечислена реализация составленных схем отношений с помощью таблиц базы данных. Типы данных полей спроектированных таблиц указаны в нотации MS SQL Server. Имена таблиц и полей составлены в соответствии с правилами MS SQL Server. Созданные таблицы являются частью БД на сервере данных. Имя базы данных для последующего использования: «busmanage».

Таблица 14 – Спецификация таблицы «тип\_автобуса»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя поля** | **Тип** | **Хранение NULL** | **По умолчанию** | **Ограничения** |
| название\_типа\_  автобуса | nvarchar(50) | запрещено | нет | первичный ключ |
| общая\_вместимость | int | запрещено | нет | положительне число |
| сидячих\_мест | int | запрещено | нет | меньше или равно общая\_вместимость и положительное |

Таблица 15 – Спецификация таблицы «депо»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя поля** | **Тип** | **Хранение NULL** | **По умолчанию** | **Ограничения** |
| номер\_депо | int | запрещено | нет | первичный ключ |
| вместимость | int | запрещено | нет |  |
| количество\_техников | int | запрещено | нет | больше или равно нулю |

Таблица 16 – Спецификация таблицы «автобус»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя поля** | **Тип** | **Хранение NULL** | **По умолчанию** | **Ограничения** |
| гос\_номер | nvarchar(10) | запрещено | нет | первичный ключ |
| гаражный\_номер | nvarchar(5) | запрещено | нет |  |
| название\_типа\_автобуса | nvarchar(50) | запрещено | нет | внешний ключ для связи с таблицей тип\_автобуса |
| номер\_депо | int | запрещено | нет | внешний ключ для связи с таблицей депо |

Таблица 17 – Спецификация таблицы «маршрут»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя поля** | **Тип** | **Хранение NULL** | **По умолчанию** | **Ограничения** |
| номер\_маршрута | nvarchar(5) | запрещено | нет | первичный ключ |
| длина\_маршрута | float | разрешено | нет | положительное число |

Таблица 18 – Спецификация таблицы «остановка»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Хранение NULL | По умолчанию | Ограничения |
| ид\_остановки | int | запрещено | identity(1,1) | первичный ключ |
| название\_  остановки | nvarchar(50) | запрещено | нет |  |

Таблица 19 – Спецификация таблицы «остановка маршрута»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя поля** | **Тип** | **Хранение NULL** | **По умолчанию** | **Ограничения** |
| ид\_остановки\_маршрута | int | запрещено | identity(1,1) | первичный ключ |
| номер\_маршрута | nvarchar(5) | запрещено | нет | внешний ключ для связи с таблицей маршрут |
| ид\_остановки | int | запрещено | нет | внешний ключ для связи с таблицей остановка |

Таблица 20 – Спецификация таблицы «рейс»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя поля** | **Тип** | **Хранение NULL** | **По умолчанию** | **Ограничения** |
| номер\_путевого\_листа | nvarchar(12) | запрещено | нет | первичный ключ |
| гос\_номер | nvarchar(10) | запрещено | нет | внешний ключ для связи с таблицей автобус |
| номер\_маршрута | nvarchar(5) | запрещено | нет | внешний ключ для связи с таблицей маршрут |
| дата\_рейса | date | запрещено | getDate() |  |

Таблица 21 – Спецификация таблицы «перемещения\_по\_маршруту»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя поля** | **Тип** | **Хранение NULL** | **По умолчанию** | **Ограничения** |
| ид\_перемещения\_  по\_маршруту | int | запрещено | identity(1,1) | первичный ключ |
| ид\_остановки | int | запрещено | нет | внешний ключ для связи с таблицей остановка |

Окончание таблицы 21

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| номер\_путевого\_листа | nvarchar(12) | запрещено | нет | внешний ключ для связи с таблицей рейс |
| время\_прибытия | time | запрещено | нет | раньше чем время\_отбытия |
| время\_отбытия | time | разрешено | нет | позже чем время\_прибытия |

Таблица 22 – Спецификация таблицы «должности»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя поля** | **Тип** | **Хранение NULL** | **По умолчанию** | **Ограничения** |
| название\_должности | nvarchar(25) | запрещено | нет | первичный ключ |
| оклад | money | запрещено | нет | Положительное число |

Таблица 23 – Спецификация таблицы «работник»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя поля** | **Тип** | **Хранение NULL** | **По умолчанию** | **Ограничения** |
| ИНН\_работника | nvarchar(12) | запрещено | нет | первичный ключ |
| фамилия | nvarchar(25) | запрещено | нет |  |
| имя | nvarchar(30) | запрещено | нет |  |
| отчество | nvarchar(30) | разрешено | нет |  |
| название\_  должности | nvarchar(25) | запрещено | нет | внешний ключ для связи с таблицей должности |
| фотография | image | разрешено | нет |  |
| дата\_наема | date | запрещено | текущая дата  getDate() |  |
| дата\_увольнения | date | разрещено | нет |  |

Таблица 24 – Спецификация таблицы «работник\_на\_рейсе»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя поля** | **Тип** | **Хранение NULL** | **По умолчанию** | **Ограничения** |
| ид\_смены | int | запрещено | identity(1,1) | первичный ключ |
| номер\_путевого\_листа | nvarchar(12) | запрещено | нет | внешний ключ для связи с таблицей рейс |
| ИНН\_работника | nvarchar(12) | запрещено | нет | внешний ключ для связи с таблицей работник |

Структура базы данных приведена на следующей ER-диаграмме (см. рисунок 25), которая отображает таблицы базы данных, структуру таблиц базы данных и связи между ними.

Для создания базы данных, соответствующей данным спецификациям и инфологической модели, приведенной в параграфе 5, необходимо выполнить на стороне сервера следующий SQL-сценарий (листинг 1).

Листинг 1 – Код SQL-сценария для создания базы данных «busmanage»

create database busmanage;

use busmanage;

-- Создать таблицу для хранения информации о типах автобусов

create table тип\_автобуса(

ид\_типа\_автобуса int identity primary key clustered,

название nvarchar(50) not null unique,

общая\_вместимость int not null check( общая\_вместимость > 0),

сидячих\_мест int not null,

constraint вместимость\_cst

check(сидячих\_мест <= общая\_вместимость and сидячих\_мест >= 0 )

)

go

-- Создать таблицу для хранения информации о депо

create table депо(

номер\_депо int primary key clustered,

вместимость int not null,

количество\_техников int not null check(количество\_техников >= 0)

)

go

-- Создать таблицу для хранения информации о автобусах

create table автобус(

гос\_номер nvarchar(10) primary key clustered check(len(гос\_номер)>=6),

гаражный\_номер nvarchar(5),

ид\_типа\_автобуса int not null

foreign key references тип\_автобуса(ид\_типа\_автобуса),

номер\_депо int not null

foreign key references депо(номер\_депо)

)

go

-- Создать таблицу для хранения информации о маршрутах

create table маршрут(

номер\_маршрута nvarchar(5) primary key

clustered check(len(номер\_маршрута)>0),

длина\_маршрута float check( длина\_маршрута > 0 )

)

go

-- Создать таблицу для хранения информации о остановках

create table остановка(

ид\_остановки int identity primary key clustered,

название\_остановки nvarchar(50) not null

)

go

-- Создать таблицу для хранения информации о остановках маршрутов

create table остановка\_маршрута(

ид\_остановки\_маршрута int identity

primary key clustered,

номер\_маршрута nvarchar(5) not null

foreign key references маршрут(номер\_маршрута),

ид\_остановки int not null

foreign key references останов-ка(ид\_остановки)

)

go

-- Создать таблицу для хранения информации о рейсах

create table рейс(

номер\_путевого\_листа nvarchar(12)

primary key clustered check(len(номер\_путевого\_листа)=12),

гос\_номер nvarchar(10) not null

foreign key references автобус (гос\_номер),

номер\_маршрута nvarchar(5) not null

foreign key references маршрут (номер\_маршрута),

дата\_рейса DATE not null default (getDate())

)

go

-- Создать таблицу для хранения информации о перемещниях по маршрутам

create table перемещение\_по\_маршруту(

ид\_перемещение\_маршруту int

identity primary key clustered,

ид\_остановки int not null

foreign key references останов-ка(ид\_остановки),

номер\_путевого\_листа nvarchar(12) not null

foreign key references рейс(номер\_путевого\_листа),

время\_прибытия time not null,

время\_отбытия time,

constraint время\_cst check(время\_прибытия < время\_отбытия)

)

go

-- Создать таблицу для хранения информации о должностях

create table должности(

ид\_должности int identity primary key clustered,

название nvarchar(25) not null check(len(название)>0),

оклад money not null check(оклад > 0)

)

go

-- Создать таблицу для хранения информации о работниках

create table работник(

ИНН\_работника nvarchar(12)

primary key clustered check(len(ИНН\_работника)=12),

фамилия nvarchar(25) not null,

имя nvarchar(25) not null,

отчество nvarchar(25),

ид\_должности int not null

foreign key references должно-сти(ид\_должности),

фотография image,

дата\_наема DATE not null default(getDate()),

дата\_увольнения DATE

)

go

-- Создать таблицу для хранения информации о работниках на рейсе

create table работник\_на\_рейсе(

ид\_смены int identity primary key clustered,

номер\_путевого\_листа nvarchar(12) not null

foreign key references рейс (номер\_путевого\_листа),

ИНН\_работника nvarchar(12) not null

foreign key references работник (ИНН\_работника)

)

Go

**5 Хранимые процедуры для вставки данных**

В целях упрощения программного доступа к разрабатываемой базе данных и обеспечения целостности данных следует разработать SQL-команды для осуществления вставки данных в таблицы и сохранить их на сервере для последующего использования. Для этого разработаны хранимые процедуры на сервере MS SQL Server , которые могут быть вызваны всякий раз в случае необходимости добавления новых записей.

Для каждой таблицы базы данных создана хранимая процедура вставки, которая может быть использована для выполнения запроса на добавление новых записей в таблицу. Все процедуры вставки возвращают текстовый результат, который удобно вывести в панель сообщений в разрабатываемой программе управления базой данных.

**5.1 Хранимая процедура для таблицы «тип\_автобуса»**

*Имя процедуры*: Вставить\_тип\_автобуса

*Назначение процедуры*: добавление типа автобуса.

Таблица 25 – Аргументы процедуры «добавить\_тип\_источника»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Аргумент** | **Смысл** | **Вид аргумента** | **Значение по умолчанию** | **Тип** |
| @result | результат работы процедуры | выходной | 0 | nvarchar(50) |
| @назв | Название типа | входной | нет | nvarchar(50) |
| @ОВ | Общая вместимость автобуса данного типа | входной | нет | int |
| @СМ | Количество сидячих мест в автобусах данного типа | входной | нет | int |

Возвращаемый результат процедуры:

* 'Ошибка: такой тип уже есть’
* 'Ошибка: общ. вмес. менш. чем сид. мест.'
* 'Ошибка: Вставка в таблицу неудалась'
* 'Сообщение: Успех процедуры'

*Листинг 2 – SQL код создания и использования процедуры «вставить\_тип\_автобуса»*

create procedure Вставить\_тип\_автобуса

@Result nvarchar(50) output,

@Назв nvarchar(50),

@ОВ int,

@СМ int

AS

BEGIN

-- Если существует запись

if exists(select \* from тип\_автобуса where название\_типа\_автобуса = @Назв)

BEGIN

select 'Ошибка: такой тип уже есть: ' + @Назв

return

END

-- Если не удовлетворяет условию целостности

-- Общих мест меньше чем сидячих и сидячих мест меньше нуля

if @ОВ < @СМ and @СМ>=0

BEGIN

select 'Ошибка: общ. вмес. менш. чем сид. мест. В ' + @Назв

return

END

INSERT INTO тип\_автобуса(название\_типа\_автобуса, общая\_вместимость, сидячих\_мест) values (@Назв, @ОВ, @СМ)

-- Проверка на успешную вставку

if @@ROWCOUNT > 0

BEGIN

select 'Сообщение: Успех процедуры'

END

ELSE

BEGIN

select 'Ошибка: Вставка в таблицу неудалась'

END

return

END

GO

*--Пример использования:*

declare @res nvarchar(50)

exec Вставить\_тип\_автобуса @res output, 'МТ-10', 30, 20

select @res

**5.2 Хранимая процедура для таблицы «депо»**

*Имя процедуры*: Вставить\_депо

*Назначение*: Добавление новой записи в таблицу депо

Таблица 26 – Аргументы процедуры «добавить\_страну»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Аргумент** | **Смысл** | **Вид аргумента** | **Значение**  **по умолчанию** | **Тип** |
| @result | результат работы процедуры | выходной | 0 | nvarchar(50) |
| @номер | Номер гаража на территории предприятия | входной | нет | int |
| @вместимость | количество мест для размещения техники | входной | нет | int |
| @техники | Количество технического персонала | входной | нет | int |

Возвращаемый результат процедуры:

* 'Уже есть депо с таким номером'
* 'Сообщение: Успех процедуры'
* 'Ошибка: Вставка в таблицу неудалась'

*Листинг 3 – SQL код создания и использования процедуры «добавить\_страну»*

create procedure Вставить\_депо

@OUT nvarchar(50) output,

@номер int,

@вместимость int,

@техники int

AS

BEGIN

if exists(select \* from депо where номер\_депо = @номер)

BEGIN

select 'Уже есть депо с таким номером'

return

END

insert into депо(номер\_депо, вместимость, количество\_техников) values (@номер, @вместимость, @техники)

if @@ROWCOUNT > 0

BEGIN

select 'Сообщение: Успех процедуры'

return

END

ELSE

BEGIN

select 'Ошибка: Вставка в таблицу неудалась'

END

return

END

GO

*--Пример использования:*

declare @res nvarchar(50)

exec Вставить\_депо @res output, 101, 10, 5

select @res

**5.3 Хранимая процедура для таблицы «автобус»**

*Имя процедуры*: добавить\_автобус

*Назначение процедуры*: добавление информации об автобусе.

*Требования к целостности данных*: название типа автобуса и номер депо должны существовать в таблицах «тип\_автобуса» и в таблице «депо».

Таблица 27 – Аргументы процедуры «добавить\_автобус»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Аргумент** | **Смысл** | **Вид аргумента** | **Значение**  **по умолчанию** | **Тип** |
| @result | результат работы  процедуры | выходной | 0 | nvarchar(50) |
| @Гос\_номер | Государственный регистрационный номер транспорного средства | входной | нет | nvarchar(10) |
| @Гар\_номер | Гаражный номер транспортного средства | входной | нет | nvarchar(5) |
| @тип | Тип автобуса | входной | нет | nvarchar(50) |
| @Номер депо | Номер депо к которому приписан автобус | входной | null | int |

Возвращаемый результат процедуры:

* 'Ошибка: такой автобус уже есть: '
* 'Ошибка: нет такого типа автобуса: '
* 'Ошибка: нет такого депо: '
* 'Сообщение: успех процедуры'
* 'Ошибка: Вставка в таблицу неудалась'

*Листинг 4 – SQL код создания и использования процедуры «Вставить\_автобус»*

create procedure Вставить\_автобус

@OUT nvarchar(50) output,

@Гос\_Номер nvarchar(10),

@Гар\_Номер nvarchar(5),

@тип nvarchar(50),

@Номер\_депо int

AS

BEGIN

if exists(select \* from автобус where гос\_номер = @Гос\_Номер)

BEGIN

select 'Ошибка: такой автобус уже есть: ' + str(@Гос\_Номер)

return

END

if not exists(select \* from тип\_автобуса where название\_типа\_автобуса = @тип)

BEGIN

Select 'Ошибка: нет такого типа автобуса: ' + @тип

return

END

if not exists(select \* from депо where номер\_депо = @Номер\_депо)

BEGIN

select 'Ошибка: нет такого депо: ' + str(@Номер\_депо)

return

END

insert into автобус(гос\_номер, гаражный\_номер, название\_типа\_автобуса, номер\_депо) values(@Гос\_Номер, @Гар\_Номер, @тип, @Номер\_депо)

if @@ROWCOUNT > 0

BEGIN

select 'Сообщение: успех процедуры'

return

END

ELSE

BEGIN

select 'Ошибка: Вставка в таблицу неудалась'

END

return

END

GO

*--Пример использования:*

declare @res nvarchar(50)

exec добавить\_автобус @res output, ‘К190УМ’, '502А','МГ-10', 101

select @res

**5.4 Хранимая процедура для таблицы «должность»**

*Имя процедуры*: вставить\_должность

*Назначение процедуры*: добавление информации об должностях в организации.

Таблица 28 – Аргументы процедуры «Вставить\_должность»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Аргумент** | **Смысл** | **Вид аргумента** | **Значение**  **по умолчанию** | **Тип** |
| @result | результат работы  процедуры | выходной | 0 | nvarchar(50) |
| @название | Название должности | входной | нет | nvarchar(25) |
| @Оклад | Оклад работника на должности | входной | нет | money |

Возвращаемый результат процедуры:

* 'Ошибка: уже есть такая должность'
* 'Сообщение: успех процедуры'
* 'Ошибка: Вставка в таблицу неудалась'

*Листинг 5 – SQL код создания и использования процедуры «Вставить\_должность»*

create procedure Вставить\_должность

@OUT nvarchar(50) output,

@Название nvarchar(25),

@Оклад money

AS

BEGIN

if exists(select \* from должности where название\_должности = @Название)

BEGIN

select 'Ошибка: уже есть такая должность'

return

END

insert into должности(название\_должности, оклад) values(@Название, @Оклад)

if @@ROWCOUNT > 0

BEGIN

select 'Сообщение: успех процедуры'

return

END

ELSE

BEGIN

select 'Ошибка: Вставка в таблицу неудалась'

END

END

GO

—пример исполь*зования:*

declare @res nvarchar(50)

exec Вставить\_должность @res output, 'Водитель 1-ктг', 30000

select @res

**5.5 Хранимая процедура для таблицы «работник»**

*Имя процедуры*: Вставить\_работника

*Назначение процедуры*: добавление информации о работнике

Таблица 29– Аргументы процедуры «Вставить\_работника»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Аргумент** | **Смысл** | **Вид аргумента** | **Значение**  **по умолчанию** | **Тип** |
| @result | результат работы  процедуры | выходной | 0 | nvarchar(50) |
| @ИНН | ИНН работника | входной | нет | nvarchar(12) |
| @Фамилия | Фамилия работника | входной | нет | nvarchar(25) |
| @Имя | Имя работника | входной | нет | nvarchar(25) |
| @Отчество | Отчество работника | входной | нет | nvarchar(25) |
| @Должность | Должность занимаемая работником | входной | нет | nvarchar(25) |
| @Фотография | Фотография работника | входной | нет | image |
| @Дата\_наема | Дата приема работника на работу | входной | нет | date |
| @дата\_увольнения | Дата увольнения работника с работы | входной | нет | date |

Возвращаемый результат процедуры:

* 'Ошибка: Работник с этим ИНН уже есть'
* 'Ошибка: ИНН не верен (!=12 цифр)'
* 'Ошибка: нет такой должности'
* 'Ошибка: дата наема позже даты увольнения'

*Листинг 6 – SQL код создания и использования процедуры «вставить\_работника»*

create procedure Вставить\_работника

@OUT nvarchar(50) output,

@ИНН nvarchar(12),

@Фамилия nvarchar(25),

@Имя nvarchar(25),

@Отчество nvarchar(25),

@должность nvarchar(25),

@Фотография image,

@дата\_наема date,

@дата\_увольнения date

AS

BEGIN

if exists(select \* from работник where ИНН\_работника = @ИНН)

BEGIN

select 'Ошибка: Работник с этим ИНН уже есть'

return

END

if LEN(@ИНН)!=12

BEGIN

select 'Ошибка: ИНН не верен (!=12 цифр)'

return

END

if @дата\_увольнения is not null

BEGIN

if @дата\_наема > @дата\_увольнения

BEGIN

select 'Ошибка: дата наема позже даты увольнения'

return

END

END

if not exists(select \* from должности where название\_должности = @должность)

BEGIN

select 'Ошибка: нет такой должности'

return

END

insert into работник(ИНН\_работника, фамилия, имя, отчество, название\_должности, фотография, дата\_наема, дата\_увольнения)

values (@ИНН, @Фамилия, @Имя, @Отчество, @должность, @Фотография, @дата\_наема, @дата\_увольнения)

END

GO

--Пример использования:

declare @res nvarchar(50)

exec Вставить\_работника @out, '145785963784', 'Пистолетов', 'Анатолий', 'Михаилович', 'Контролер', null, '20170506' , null

select @res

**5.6 Хранимая процедура для таблицы «остановка»**

*Имя процедуры*: Вставить\_остановка

*Назначение процедуры*: добавление информации об остановках.

Таблица 30 – Аргументы процедуры «добавить\_представителя\_стиля»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Аргумент** | **Смысл** | **Вид аргумента** | **Значение**  **по умолчанию** | **Тип** |
| @result | результат работы  процедуры | выходной | 0 | nvarchar(50) |
| @Название\_остановки | Название остановочного пункта | входной | нет | nvarchar(50) |

Возвращаемый результат процедуры:

* 'Сообщение: Успех процедуры'
* 'Ошибка: Вставка в таблицу неудалась'

*Листинг 7 – SQL код создания и использования процедуры «Вставить\_остановка»*

create procedure Вставить\_остановка

@OUT nvarchar(50) output,

@Название\_остановки nvarchar(50)

AS

BEGIN

insert into остановка(название\_остановки) values(@Название\_остановки)

if @@ROWCOUNT > 0

BEGIN

select 'Сообщение: Успех процедуры'

return

END

ELSE

BEGIN

select 'Ошибка: Вставка в таблицу неудалась'

END

return

END

GO

--Пример использования:

declare @res nvarchar(50)

exec Вставить\_остановка @res output, 'Кинотеатр'

select @res

**5.7 Хранимая процедура для таблицы «Вставить\_маршрут»**

*Имя процедуры*: Вставить\_маршрут

*Назначение процедуры*: добавление информации о маршрутах

Таблица 31 – Аргументы процедуры «Вставить\_маршрут»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Аргумент** | **Смысл** | **Вид аргумента** | **Значение**  **по умолчанию** | **Тип** |
| @result | результат работы  процедуры | выходной | 0 | nvarchar(50) |
| @Номер | Номер маршрута | входной | нет | nvarchar(5) |
| @Длина | Длина маршрута | входной | нет | float |

Возвращаемый результат процедуры:

* 'Ошибка: Такой маршрут уже есть'
* 'Сообщение: Успех процедуры'
* 'Ошибка: Вставка в таблицу неудалась'

*Листинг 8 – SQL код создания и использования процедуры «Вставить\_маршрут»*

create procedure Вставить\_маршрут

@OUT nvarchar(50) output,

@Номер nvarchar(5),

@длина float

AS

BEGIN

if exists(select \* from маршрут where номер\_маршрута = @Номер)

BEGIN

select 'Ошибка: Такой маршрут уже есть'

return

END

insert into маршрут(номер\_маршрута, длина\_маршрута) values(@Номер, @длина)

if @@ROWCOUNT > 0

BEGIN

select 'Сообщение: Успех процедуры'

return

END

ELSE

BEGIN

select 'Ошибка: Вставка в таблицу неудалась'

END

return

END

GO

Пример использования:

declare @res nvarchar(50)

exec Вставить\_маршрут @out, '105a', 13

select @res

**5.8 Хранимая процедура для таблицы «остановка маршрута»**

*Имя процедуры*: Вставить\_остановка\_маршрута

*Назначение процедуры*: Вставка информации об остановке маршрута

Таблица 32 – Аргументы процедуры «Вставить\_остановка\_маршрута»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Аргумент** | **Смысл** | **Вид аргумента** | **Значение**  **по умолчанию** | **Тип** |
| @result | результат работы  процедуры | выходной | 0 | nvarchar(50) |
| @номер\_маршрута | номер маршрута движения автобусов | входной | нет | nvarchar(5) |
| @остановка | индификатор остановки | входной | null | int |

Возвращаемый результат процедуры:

* 'Ошибка: такая остановка маршрута уже есть'
* 'Ошибка: нет такой остановки'
* 'Ошибка: нет такого маршрута'
* 'Сообщение: Успех процедуры'
* 'Ошибка: Вставка в таблицу неудалась'

*Листинг 9 – SQL код создания и использования процедуры «добавить\_материал»*

create procedure Вставить\_остановку\_маршрута

@OUT nvarchar(50) output,

@номер\_маршрута nvarchar(5),

@остановка int

AS

BEGIN

if exists(select \* from остановка\_маршрута where номер\_маршрута = @номер\_маршрута

and ид\_остановки = @остановка)

BEGIN

select 'Ошибка: такая остановка маршрута уже есть'

return

END

if not exists(select \* from остановка where ид\_остановки = @остановка)

BEGIN

select 'Ошибка: нет такой остановки'

return

END

if not exists(select \* from маршрут where номер\_маршрута = @номер\_маршрута)

BEGIN

select 'Ошибка: нет такого маршрута'

return

END

insert into остановка\_маршрута(номер\_маршрута,ид\_остановки) values(@номер\_маршрута, @остановка)

if @@ROWCOUNT > 0

BEGIN

select 'Сообщение: Успех процедуры'

return

END

ELSE

BEGIN

select 'Ошибка: Вставка в таблицу неудалась'

END

return

END

GO

Пример использования:

declare @res nvarchar(50)

exec Вставить\_остановку\_маршрута @out, '62', 5

select @res

**5.9 Хранимая процедура для таблицы «Рейс»**

*Имя процедуры*: Вставить\_рейс

*Назначение процедуры*: Добавление информации о рейсе

Таблица 33 – Аргументы процедуры «Вставить\_рейс»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Аргумент** | **Смысл** | **Вид аргумента** | **Значение**  **по умолчанию** | **Тип** |
| @result | результат работы  процедуры | выходной | 0 | nvarchar(50) |
| @номер\_пут\_листа | Номер путевого листа рейса | входной | нет | nvarchar(12) |
| @гос\_номер\_автобуса | Номер автобуса совершающего рейс | входной | нет | nvarchar(10) |
| @номер маршрута | Номер маршрута по которому совершаеться рейс | входной | нет | nvarchar(5) |
| @Дата\_Рейса | Дата совершения реса | входной | нет | date |

Возвращаемый результат процедуры:

* 'Ошибка: Рейс с таким НПЛ уже есть'
* 'Ошибка: автобуса с таким номером нет'
* 'Ошибка: такого маршрута нет'
* 'Сообщение: Успех процедуры'
* 'Ошибка: Вставка в таблицу неудалась'

*Листинг 10 – SQL код создания и использования процедуры «Вставить\_рейс»*

create procedure Вставить\_Рейс

@OUT nvarchar(50) output,

@номер\_пут\_листа nvarchar(12),

@гос\_номер\_автобуса nvarchar(10),

@номер\_маршрута nvarchar(5),

@Дата\_рейса DATE

AS

BEGIN

if exists(select \* from рейс where номер\_путевого\_листа = @номер\_пут\_листа)

BEGIN

select 'Ошибка: Рейс с таким НПЛ уже есть'

return

END

if not exists(select \* from автобус where гос\_номер = @гос\_номер\_автобуса)

BEGIN

select 'Ошибка: автобуса с таким номером нет'

return

END

if not exists(select \* from маршрут where номер\_маршрута = @номер\_маршрута)

BEGIN

select 'Ошибка: такого маршрута нет'

return

END

insert into рейс(номер\_путевого\_листа, гос\_номер, номер\_маршрута, дата\_рейса)

values (@номер\_пут\_листа, @гос\_номер\_автобуса, @номер\_маршрута, @Дата\_рейса)

if @@ROWCOUNT > 0

BEGIN

select 'Сообщение: Успех процедуры'

return

END

ELSE

BEGIN

select 'Ошибка: Вставка в таблицу неудалась'

END

return

END

GO

--Пример использования:

declare @res nvarchar(50)

exec Вставить\_Рейс @out, '738-11AV', 'К707ЕК65', '62', '20180404'

select @res

**5.10 Хранимая процедура для таблицы «перемещение по маршруту»**

*Имя процедуры*: Вставить\_перемещение\_по\_маршруту

*Назначение процедуры*: Добавление данных о перемещениях по маршруту

*Требования к целостности данных*: дата прибытия должна быть раньше чем дата отбытия

Таблица 34 – Аргументы процедуры «Вставить\_перемещение\_по\_маршруту»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Аргумент** | **Смысл** | **Вид аргумента** | **Значение**  **по умолчанию** | **Тип** |
| @result | результат работы  процедуры | выходной | 0 | nvarchar(50) |
| @Остановка | индификатор остановки | входной | нет | int |
| @номер\_пут\_листа | номер путевого листа по которому идет перемещение | входной | нет | nvarchar(12) |
| @Время\_прибытия | Время прибытия на остановочный пункт | входной | нет | TIME |
| @время отбытия | Время отбытия с остановочного пункта | входной | нет | TIME |

Возвращаемый результат процедуры:

* 'Ошибка: нет остановки'
* 'Ошибка: нет маршрута'
* 'Ошибка: время прибытия позже отбытия'
* 'Сообщение: Успех процедуры'
* 'Ошибка: Вставка в таблицу неудалась'

*Листинг 11 – SQL код создания и использования процедуры «Вставить\_перемещение\_по\_маршруту»*

create procedure Вставить\_перемещение\_по\_маршруту

@OUT nvarchar(50) output,

@остановка int,

@номер\_пут\_листа nvarchar(12),

@Время\_прибытия TIME,

@время\_отбытия TIME

AS

BEGIN

if not exists(select \* from остановка where ид\_остановки = @остановка)

BEGIN

select 'Ошибка: нет остановки'

return

END

if not exists(select \* from рейс where номер\_путевого\_листа = @номер\_пут\_листа)

BEGIN

select 'Ошибка: нет маршрута'

return

END

if @время\_отбытия is not null

BEGIN

if @Время\_прибытия > @время\_отбытия

BEGIN

select 'Ошибка: время прибытия позже отбытия'

return

END

END

insert into перемещение\_по\_маршруту(ид\_остановки, номер\_путевого\_листа, время\_прибытия, время\_отбытия)

values(@остановка, @номер\_пут\_листа, @Время\_прибытия, @время\_отбытия)

if @@ROWCOUNT > 0

BEGIN

select 'Сообщение: Успех процедуры'

return

END

ELSE

BEGIN

select 'Ошибка: Вставка в таблицу неудалась'

END

return

END

GO

--Пример использования (нулевая дата найма):

declare @res nvarchar(50)

exec Вставить\_перемещение\_по\_маршруту @out, 1, '738-11AV', '1255', ‘1310’

select @res

**5.11 Хранимая процедура для таблицы «работник на рейсе»**

*Имя процедуры*: Вставить\_работника\_на\_рейсе

*Назначение процедуры*: Добавление информации о работниках на рейсе

*Требования к целостности данных*: ИНН работника и номер путевого листа должны существовать в таблицах «работник» и «рейс» соответственно.

Таблица 35 – Аргументы процедуры «Вставить\_работника\_на\_рейсе»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Аргумент** | **Смысл** | **Вид аргумента** | **Значение**  **по умолчанию** | **Тип** |
| @result | результат работы | выходной | 0 | nvarchar(50) |
| @номер\_пут\_листа | Номер путевого листа рейса | входной | нет | nvarchar(12) |
| @ИНН\_работника | ИНН работника работающего на рейсе | входной | нет | nvarchar(12) |

Возвращаемый результат процедуры:

* 'Ошибка: нет такого рейса'
* 'Ошибка: нет такого работника'
* 'Сообщение: Успех процедуры'
* 'Ошибка: Вставка в таблицу неудалась'

*Листинг 12 – SQL код создания и использования процедуры «Вставить\_работника\_на\_рейсе»*

create procedure Вставить\_работника\_на\_рейсе

@OUT nvarchar(50) output,

@номер\_пут\_листа nvarchar(12),

@ИНН\_Рабтника nvarchar(12)

AS

BEGIN

if not exists(select \* from рейс where номер\_путевого\_листа = @номер\_пут\_листа)

BEGIN

select 'Ошибка: нет такого рейса'

return

END

if not exists(select \* from работник where ИНН\_работника = @ИНН\_Рабтника)

BEGIN

select 'Ошибка: нет такого работника'

return

END

insert into работник\_на\_рейсе(номер\_путевого\_листа, ИНН\_работника) values (@номер\_пут\_листа, @ИНН\_Рабтника)

if @@ROWCOUNT > 0

BEGIN

select 'Сообщение: Успех процедуры'

return

END

ELSE

BEGIN

select 'Ошибка: Вставка в таблицу неудалась'

END

return

END

GO

--Пример использования:

declare @res nvarchar(50)

exec Вставить\_работника\_на\_рейсе @out, '758-11AV', '145785963784'

select @res

*Листинг 13 – SQL код заполнения базы данных данными*

*use busmanage;*

*Declare @out nvarchar(50)*

*exec Вставить\_депо @out, 100, 5, 5*

*exec Вставить\_депо @out, 200, 15, 5*

*exec Вставить\_депо @out, 300, 10, 4*

*exec Вставить\_депо @out, 400, 10, 2*

*exec Вставить\_депо @out, 500, 7, 3*

*exec Вставить\_депо @out, 600, 10, 4*

*exec Вставить\_депо @out, 700, 15, 4*

*exec Вставить\_депо @out, 701, 5, 2*

*exec Вставить\_депо @out, 800, 10, 3*

*exec Вставить\_депо @out, 900, 5, 0*

*exec Вставить\_тип\_автобуса @out, 'ТК-22', 30, 20*

*exec Вставить\_тип\_автобуса @out, 'ТК-11', 40, 20*

*exec Вставить\_тип\_автобуса @out, 'ГАРМОНЬ', 40, 15*

*exec Вставить\_тип\_автобуса @out, 'ТУРИСТ-5', 20, 0*

*exec Вставить\_тип\_автобуса @out, 'ГАОН-8', 60, 40*

*exec Вставить\_тип\_автобуса @out, 'ТА-5', 30, 0*

*exec Вставить\_тип\_автобуса @out, 'ОБ-12', 30, 30*

*exec Вставить\_тип\_автобуса @out, 'ГРОм', 20, 10*

*exec Вставить\_тип\_автобуса @out, 'ТК-6', 35, 20*

*exec Вставить\_тип\_автобуса @out, 'МОРж', 30, 20*

*exec Вставить\_автобус @out, 'К127УМ65', '100A', 'ТК-22', 100*

*exec Вставить\_автобус @out, 'Д120УГ65', '100Б', 'ТК-11', 100*

*exec Вставить\_автобус @out, 'К108ИМ65', '200A', 'ТК-6', 200*

*exec Вставить\_автобус @out, 'С820ОГ65', '200Б', 'ТА-5', 200*

*exec Вставить\_автобус @out, 'А463УШ65', '200В', 'ТУРИСТ-5', 200*

*exec Вставить\_автобус @out, 'М102УГ65', '800A', 'ГАРМОНЬ', 800*

*exec Вставить\_автобус @out, 'К124УН65', '300A', 'ГРОм', 300*

*exec Вставить\_автобус @out, 'Т015ЯН65', '300Б', 'ТК-6', 300*

*exec Вставить\_автобус @out, 'К707ЕК65', '300В', 'МОРж', 300*

*exec Вставить\_автобус @out, 'О893РУ65', '300Г', 'ТК-11', 300*

*exec Вставить\_должность @out, 'Водитель 1-ктг', 30000*

*exec Вставить\_должность @out, 'Водитель 2-ктг', 35000*

*exec Вставить\_должность @out, 'Водитель 3-ктг', 40000*

*exec Вставить\_должность @out, 'Водитель 4-ктг', 43000*

*exec Вставить\_должность @out, 'Кондуктор', 20000*

*exec Вставить\_должность @out, 'Контролер', 22000*

*exec Вставить\_должность @out, 'Слесарь-механик', 25000*

*exec Вставить\_должность @out, 'Механик', 27000*

*exec Вставить\_должность @out, 'Старший механик', 30000*

*exec Вставить\_должность @out, 'Инженер-механик', 31000*

*exec Вставить\_работника @out, '142548963784', 'Пупкин', 'Василий', 'Домоклович', 'Водитель 1-ктг', null, '20180506' , null*

*exec Вставить\_работника @out, '141844963784', 'Горохов', 'Александр', 'Васильевич', 'Водитель 2-ктг', null, '20130506' , '20180211'*

*exec Вставить\_работника @out, '142548945784', 'Окорочков', 'Дмитрий', 'Валерьевич', 'Водитель 3-ктг', null, '20150506' , null*

*exec Вставить\_работника @out, '142555124784', 'Горшков', 'Николай', 'Акакеевич', 'Водитель 1-ктг', null, '20130506' , null*

*exec Вставить\_работника @out, '114888963784', 'Недежурин', 'Василий', 'Григореевич', 'Водитель 1-ктг', null, '20150506' , null*

*exec Вставить\_работника @out, '142148823784', 'Ожурков', 'Геннадий', 'Геннадиевич', 'Водитель 4-ктг', null, '20130506' , null*

*exec Вставить\_работника @out, '142548965214', 'Ожуров', 'Пётр', 'Васильевич', 'Механик', null, '20120506' , null*

*exec Вставить\_работника @out, '599948963784', 'Ощуп', 'Наталья', 'Павловна', 'Кондуктор', null, '20180506' , null*

*exec Вставить\_работника @out, '145785963784', 'Пистолетов', 'Анатолий', 'Михаилович', 'Контролер', null, '20170506' , null*

*exec Вставить\_работника @out, '189154893784', 'Коддратьев', 'Михаил', null, 'Водитель 2-ктг', null, '20130506' , null*

*exec Вставить\_остановка @out, 'Больничная'*

*exec Вставить\_остановка @out, 'Парк отдыха'*

*exec Вставить\_остановка @out, 'Дом торговли'*

*exec Вставить\_остановка @out, 'Кинотеатр'*

*exec Вставить\_остановка @out, 'Детский сад'*

*exec Вставить\_остановка @out, 'Телецентр'*

*exec Вставить\_остановка @out, 'Фабрика'*

*exec Вставить\_остановка @out, 'Поляново'*

*exec Вставить\_остановка @out, 'Дом офицеров'*

*exec Вставить\_остановка @out, 'Овощебаза'*

*exec Вставить\_остановка @out, 'Университет'*

*exec Вставить\_маршрут @out, '105', 5.7*

*exec Вставить\_маршрут @out, '105a', 13*

*exec Вставить\_маршрут @out, '1', 5.7*

*exec Вставить\_маршрут @out, '1a', 6*

*exec Вставить\_маршрут @out, '1b', 5.7*

*exec Вставить\_маршрут @out, '62', 5.7*

*exec Вставить\_маршрут @out, '32', 8*

*exec Вставить\_маршрут @out, '11a', 4*

*exec Вставить\_маршрут @out, '1c', 5.0*

*exec Вставить\_маршрут @out, '71', 3*

*exec Вставить\_остановку\_маршрута @out, '1', 11*

*exec Вставить\_остановку\_маршрута @out, '1', 10*

*exec Вставить\_остановку\_маршрута @out, '71', 3*

*exec Вставить\_остановку\_маршрута @out, '71', 8*

*exec Вставить\_остановку\_маршрута @out, '1b', 2*

*exec Вставить\_остановку\_маршрута @out, '1b', 3*

*exec Вставить\_остановку\_маршрута @out, '1b', 4*

*exec Вставить\_остановку\_маршрута @out, '62', 2*

*exec Вставить\_остановку\_маршрута @out, '62', 3*

*exec Вставить\_остановку\_маршрута @out, '62', 1*

*exec Вставить\_остановку\_маршрута @out, '62', 5*

*exec Вставить\_Рейс @out, '758-96AE', 'О893РУ65', '71', '20180507'*

*exec Вставить\_Рейс @out, '758-11AV', 'К707ЕК65', '62', '20180418'*

*exec Вставить\_Рейс null, '778-96AE', 'М102УГ65', '1a', '20180508'*

*exec Вставить\_Рейс @out, '758-9E', 'К124УН65', '105a', '20180512'*

*exec Вставить\_Рейс @out, '758-96A', 'О893РУ65', '71', '20180515'*

*exec Вставить\_Рейс @out, '758-11V', 'К707ЕК65', '62', '20180419'*

*exec Вставить\_Рейс @out, '7118-96AE', 'М102УГ65', '1a', '20180511'*

*exec Вставить\_Рейс @out, '718-9E', 'К124УН65', '105a', '20180517'*

*exec Вставить\_Рейс @out, '778-96AE', 'О893РУ65', '71', '20180507'*

*exec Вставить\_Рейс @out, '738-11AV', 'К707ЕК65', '62', '20180404'*

*exec Вставить\_работника\_на\_рейсе @out, '738-11AV', '142548963784'*

*exec Вставить\_работника\_на\_рейсе @out, '738-11AV', '599948963784'*

*exec Вставить\_работника\_на\_рейсе @out, '738-11AV', '145785963784'*

*exec Вставить\_работника\_на\_рейсе @out, '778-96AE', '189154893784'*

*exec Вставить\_работника\_на\_рейсе @out, '738-11AV', '145785963784'*

*exec Вставить\_работника\_на\_рейсе @out, '758-9E', '142555124784'*

*exec Вставить\_работника\_на\_рейсе @out, '758-96AE', '142148823784'*

*exec Вставить\_работника\_на\_рейсе @out, '758-11AV', '142148823784'*

*exec Вставить\_работника\_на\_рейсе @out, '718-9E', '145785963784'*

*exec Вставить\_работника\_на\_рейсе @out, '738-11AV', '145785963784'*

*exec Вставить\_работника\_на\_рейсе @out, '758-11AV', '145785963784'*

*Declare @out nvarchar(50)*

*exec Вставить\_перемещение\_по\_маршруту @out, 1, '738-11AV', '1208', null*

*exec Вставить\_перемещение\_по\_маршруту @out, 2, '738-11AV', '1218', null*

*exec Вставить\_перемещение\_по\_маршруту @out, 3, '738-11AV', '1225', null*

*exec Вставить\_перемещение\_по\_маршруту @out, 5, '738-11AV', '1240', null*

*exec Вставить\_перемещение\_по\_маршруту @out, 1, '738-11AV', '1255', null*

*exec Вставить\_перемещение\_по\_маршруту @out, 7, '778-96AE', '1015', null*

*exec Вставить\_перемещение\_по\_маршруту @out, 8, '778-96AE', '1035', null*

*exec Вставить\_перемещение\_по\_маршруту @out, 7, '778-96AE', '1100', null*

*exec Вставить\_перемещение\_по\_маршруту @out, 8, '778-96AE', '1125', null*

*exec Вставить\_перемещение\_по\_маршруту @out, 7, '778-96AE', '1140', null*

**6 Представления на выборку данных из базы данных**

В целях ускорения процесса последующей разработки программного обеспечения и облегчения выборки данных для созданной базы данных разработаны представления, осуществляющие выборку информации *из связанных таблиц*.

**6.1 Представления выборки из одиночных таблиц**

Данная часть представлений проектируется для представления одной таблицы в программе, замещая бессмысленные для предметной области данные понятной пользователю информацией, например, в таблице «Работник на рейсе», заместить поле «ИНН работника», являющемся текстовым полем с 12-значным кодом – полями таблицы «Работник», возвращающим фамилию, имя и отчество работника. С точки зрения выбранного метода проектирования программного обеспечения, данный вид представлений необходим для большинства таблиц, так как, кроме прочего, упрощает возможность проектировать конечный вариант таблиц, видимых пользователю, позволяет, ещё на этапе разработки, установить ширину столбцов.

Листинг *14 – Код SQL – запроса создания представления «информация\_о\_рейсе»*

create view информация\_о\_рейсе as

select

рейс.номер\_путевого\_листа,

работник.фамилия + ' ' + работник.имя as [работник],

работник.название\_должности,

рейс.гос\_номер,

автобус.название\_типа\_автобуса,

тип\_автобуса.общая\_вместимость,

дата\_рейса

from рейс

join автобус On автобус.гос\_номер = рейс.гос\_номер

join тип\_автобуса on тип\_автобуса.название\_типа\_автобуса = автобус.название\_типа\_автобуса

join работник\_на\_рейсе on работник\_на\_рейсе.номер\_путевого\_листа = рейс.номер\_путевого\_листа

join работник on работник.ИНН\_работника = работник\_на\_рейсе.ИНН\_работника

go

--Пример вызова

Select \* from информация\_о\_рейсе

*Листинг 15 – Код SQL – запроса создания представления «информация\_о\_работниках»*

create view информация\_о\_работниках as

select

работник.фамилия + ' ' + работник.имя + ' ' +isnull(работник.отчество,'') as [Работник],

работник.название\_должности,

должности.оклад,

работник.дата\_наема,

работник.дата\_увольнения

from работник

join должности on должности.название\_должности = работник.название\_должности

go

--Пример вызова

Select \* from информация\_о\_работниках

*Листинг 16 – Код SQL – запроса создания представления «информация\_о\_перемещениях»*

create view информация\_о\_перемещениях as

select

перемещение\_по\_маршруту.ид\_перемещение\_маршруту as [запись],

остановка.название\_остановки,

перемещение\_по\_маршруту.время\_прибытия,

перемещение\_по\_маршруту.время\_отбытия,

рейс.гос\_номер,

рейс.номер\_маршрута,

рейс.дата\_рейса

from перемещение\_по\_маршруту

join остановка on остановка.ид\_остановки = перемещение\_по\_маршруту.ид\_остановки

join рейс on рейс.номер\_путевого\_листа = перемещение\_по\_маршруту.номер\_путевого\_листа

go

--Пример вызова

Select \* from информация\_о\_перемещениях

*Листинг 17 – Код SQL – запроса создания представления «информация\_о\_депо»*

create view информация\_о\_депо as

select

депо.номер\_депо,

вместимость,

количество\_техников,

автобус.гос\_номер,

автобус.название\_типа\_автобуса

from депо

join автобус on автобус.номер\_депо = депо.номер\_депо

go

--Пример вызова

Select \* from информация\_о\_депо

**7 Проектирование программного обеспечения**

Для управления базой данных пользователями было создано программное приложение на языке C++ в среде разработки Borland C++ Builder 6, с использованием компонентов Raize и технологии доступа к СУБД ADO. Программа предназначена для следующих групп пользователей:

* Администратор базы данных. Занимается поддержкой приложения, создаёт базу данных из прилагаемого кода.
* Администратор музея, кадровый менеджер. Управляет информацией о сотрудниках.
* Хозяйственный работник. Управляет экспонатами.
* Научный сотрудник. Управляет галереями.
* Реставратор. Реставрирует и оставляет соответствующие отметки о проделанной работе.

Рассмотрим некоторые операции, демонстрирующие функции программы.

**7.1 Авторизация пользователя**

Данная операция предусмотрена с целью подключения программы к серверу данных и должна быть выполнена любым пользователем программы.

**Входные данные операции**: имя сервера данных и пароль учётной записи для доступа к БД «магазин».

**Результат операции**: регистрация пользователя на сервере данных с возможностью выполнения операций по манипулированию данными.

Алгоритм реализации операции авторизации в виде диаграммы деятельности в нотации UML приводится на рисунке 26.

**7.2 Добавление экспоната на реставрацию**

Данная операция нацелена на администратора или научного сотрудника и позволяет ему зарегистрировать зявку на реставрацию экспоната.

**Входные данные операции**: название экспоната, ИД сотрудника, дата приёма на реставрацию, текущее состояние экспоната.

**Результат операции**: регистрация нового экспоната на реставрацию.

Алгоритм реализации операции добавления экспоната на реставрацию в виде диаграммы деятельности в нотации UML приведён на рисунке 27.

сервер БД

программа

пользователь

запрос имени и пароля для входа

переход на страницу «настройки»

ввод имени сервера и пароля

отправка данных на сервер

проверка учётных данных

создание соединения

отказ создания соединения

[

успешно

]

появление возможности просматривать и редактировать страницы данных

[

не успешно

]

запрос имени и пароля

выбор операций и дальнейшая работа

[

соглашение

]

[

отказ

]

Рисунок 26 – Диаграмма деятельности для операции авторизации пользователя



Рисунок 27 – Диаграмма деятельности для операции «Добавление на реставрацию»

**7.3 Реставрация экспоната**

Данная операция нацелена на реставратора и позволяет ему пометить экспонат в системе как отреставрированный.

**Входные данные операции**: выбранный экспонат на реставрацию, дата завершения реставрации, конечное состояние экспоната, стоимость реставрации (опционально).

**Результат операции**: отметка в системе о завершении реставрации.

Рассмотрим алгоритм реализации операции реставрации в системе в виде диаграммы деятельности в нотации UML.



Рисунок 28 - Диаграмма деятельности для операции «Реставрация экспоната»

**7.4 Формирование отчёта об имеющихся экспонатах**

Данная операция нацелена на администратора и позволяет ему в печатном виде посмотреть краткую информацию об имеющихся экспонатах.

**Входные данные операции**: нет.

**Результат операции**: распечатанный на бумаге документ, отображающий информацию о всех экспонатах.

**Структура документа отчёта**: название экспоната, автор, категория, материал, стиль, дата получения, год создания, источник, галерея, страна, миниатюра.

Рассмотрим алгоритм реализации операции авторизации в виде диаграммы деятельности в нотации UML.



Рисунок 29 - Диаграмма деятельности для операции «Формирование отчёта об имеющихся экспонатах»

Во время проектирования программного обеспечения было принято решение о максимальной автоматизации со стороны сервера. Таким образом, программа не проверяет входные данные, а только передаёт их серверу, получая сообщения с отчётами. Это обеспечивает бóльшую скорость обработки информации и универсальность подхода к проектированию. Вставка данных производится посредством исполнения SQL-кода вызова хранимых процедур вставки, удаление и обновление – стандартными вызовами DELETE и UPDATE. Предусмотрена возможность запретить или разрешить каскадное удаление на странице настроек.

**8 Руководство пользователя**

Разрабатоннаное программное обеспечение предназначено для выполнения операций по управлению данными для информацинной системы художетсвенного музея. В рамках программы реализованы функции для автоматизации основных операций работников данной предметной области для работы с базой данных.

**8.1 Аппаратные требования**

Информационная система имеет клиентскую и сервеную части. Серверная часть представлена сервером баз данных Microsoft SQL Server. Клиентская часть представлена разработанной автором программой. Серверная часть может размещаться как на выделенном компьютере сети, так и на локальном компьютере пользователя в случае использования без использования сети.

Разрабатываемая система предназначена для использования на персональных IBM-совместимых компьютерах с центральным Intel-совместимом процессором с частотой работы не менее 1ГГц, оперативным запоминающим устройством объёмом не менее 512 Мб, свободным дисковым пространством не менее 25 Мб для хранения программного обеспечения и 100 Мб для хранения базы данных. Обязательным условием является наличие манипулятора типа «мышь», SVGA-адаптера и монитора с разрешением не менее 1280х1024, сетевого адаптера LAN100/LAN1000, сетевого или локального принтера.

**8.2 Требования к хранилищу данных**

Для корректной работы программы требуется достаточное количество свободного места на жёстком диске, чтобы содержать изображения экспонатов. Конечный объём занимаемой памяти напрямую зависит от количества и размера изображений. Программа может работать в локальной сети, для этого достаточно иметь один сервер баз данных и ввести во время регистрации пользователя корректные данные.

Для функционирования хранилища данных на серверном или клиентском компьютере необходимо наличие функционирующей системы управления базами данных MS SQL Server (/2005/2008/2005 Express/2008 Express). В одном из функционирующих экземпляров должа быть создана база данных художественного музея (сценарий создания БД прилагается в дистрибутиве программного обеспечения). Конченому пользователю неообходимо предоставить сведения для доступа к этой базе данных: сетевое имя сервера базы данных, имя учтной записи и пароль для доступа к базе данных.

Таблица *45 – Аппаратные требования для обеспечения потребности хранилища данных*

|  |  |
| --- | --- |
| **Сервер (минимальная конфигурация)** | |
| Процессор | Intel Celeron 1000 MHz |
| Память | 512 Mb |
| Жесткий диск | 20 Gb 7200 RPM |
| **Клиент (минимальная конфигурация)** | |
| Процессор | Intel Celerom 1000 MHz |
| Память | 512 Mb |
| Жесткий диск | 6,4 Gb 5400 RPM |
| Монитор | 15" |

**8.3 Состав программного продукта**

В комплект программы входят:

* Museum.exe – исполняемый файл приложения
* config.ini – файл с настройками авторизации для автоматического входа
* SQL – папка, содержащая файлы сценариев для создания базы данных:
* create\_museum\_filled\_images.sql – Заполненная база данных с изображениями
* create\_museum\_filled.sql – Заполненная база данных без изображений
* create\_museum\_empty.sql – Пустая база данных

**8.4 Инструкции администратору**

Для работы программы требуется установленная СУБД Microsoft SQL (2005/2008/2005 Express/2008 Express). Имя сервера может быть любым, имя пользователя также может быть любым (в том числе отсутствовать, в таком случае используется имя «sa»), пароль должен обязательно присутствовать.

Во время первого запуска следует создать базу данных на сервере. Для этого, на странице «Настройки» следует ввести имя сервера, имя пользователя и пароль, затем выбрать один из предложенных вариантов базы данных в формате .sql или воспользоваться выбранным по-умолчанию, и нажать кнопку «Пересоздать базу данных». При корректно введённых данных учётной записи, на статусной панели в правом нижнем углу должен отобразиться прогресс создании базы данных. При корректном исполнении сценария, вход должен быть выполнен автоматически, и появится доступ к просмотру таблиц базы данных.

Имеется возможность запомнить имя сервера, пользователя и пароль. Если все три режима отмечены, следующий вход при запуске произойдёт автоматически.

**8.5 Инструкции актёрам предметной области**

8.5.1 Интерфейс пользователя программного обеспечения

Программа «Художественный музей» предназначена для управления базой данных художественного музея и обладает одним основным окном с множеством страниц. Среди главных страниц имеются «Настройки», «Словари», «Стержневые сущности», «Ассоциативные сущности», «Представления», «Отчёты». Переключение между главными страницами производится внизу окна. Каждая страница, кроме страницы настроек, имеет вложенные страницы, переключаемые вверху окна. В самом низу окна расположена статусная панель, возвращающая различные сообщения сервера баз данных.

На страницах таблиц в верхней части расположены элементы интерфейса, предоставляющие пользователю ввести входные данные для добавления и обновления таблиц: поля ввода строк, чисел, дат, бегунки, а также кнопки управления таблицей – «Добавить», «Удалить», и, если таблица предполагает обновление, «Обновить». В нижней половине страниц таблиц расположены таблицы, отображающие все записи выбранной таблицы.

Выбор записи осуществляется в результирующей таблице-с помощью мыши или стрелок клавиатуры. Имеется возможность упорядочивать записи по любому заголовку таблицы, щёлкнув по нему. Чтобы произвести упорядочивание наоборот, следует щёлкнуть по заголовку два раза подряд.

Изображение на странице «Экспонат» имеет контекстное меню «Открыть», «Сохранить». При левом щелчке открывается отдельное окно для просмотра изображения.

Пользователю должно быть понятно, что, сначала следует заполнить словари, затем сведенмя об основных объектах (в интерфейсе программы они названы стержневыми сущностями), ну а затем о зависимых объектах (в интерфейсе программы они названы ассоциативными).

Некоторые поля являются опциональными – добавляются в базу данных, только если отмечены галочкой. Для выбора дат следует нажать кнопку выпадающего списка, где откроется календарь.

Следует обратить внимание, что первичные ключи не могут быть обновлены, поля с первичными ключами по-особому выдалены на страницах.

Для того чтобы разрешить каскадное удаление, следует, на странице настроек, установить флажок «каскадное удаление», в противном случае будет возвращена ошибка с указанием, записи каких таблиц ссылаются на удаляемую.

На странице «Отчёты» имеется возможность вывести информацию в режим предварительного просмотра или на печать.



Рисунок 30 – Интерфейс пользователя программы «Художественный музей»

8.5.2 Роли пользователей в рамках программного обеспечения

Помимо администратора, с базой данных могут работать разные роли: реставраторы, хранители, научные сотрудники.

Администратор, или менеджер по кадрам может добавлять в систему сотрудников.

Хранитель (завхоз) управляет экспонатами и остальными данными, необходимыми для них.

Научный сотрудник делает пометки в экспонатах, к какой галерее каждый относится.

Реставратор делает пометки о завершении реставрации.

**Заключение**

Автором была разработана и реализивана концептуальная, логическая и физическая модели предметной области «Художественный музей».

Было изучено объектно-ориентированное моделирование на языке UML, проектирование и программирование баз данных на языке Transact SQL и работа с СУБД, программирование запросов к базам данных средствами компонентов ADO.

Согласно плану, была создана база данных для автоматизации учётной дяетельности художественного музея, книга поступлений реализована в виде таблицы «экспонат», присутствуют изображения.

Помими базы данных, была разработана программа, позволяющая управлять всеми таблицами базы данных, производить просмотр нескольких связанных таблиц, давая более полное представление об имеющейся информации, а также выводить отчёты на печать.

Касательно связи между программой и СУБД был выбран подход наибольшей автоматизации и слежением за целостностью данных со стороны сервера, а не клиента. Для управления базой данных была использована генерация SQL запросов вручную, без использования автоматической связи базы данных с элеменами (кроме изображения). Такой подход даёт дополнительную гибкость кода, ответственность для разработчика и возможность передавать любые сообщения из сервера после запросов без ошибок времени исполнения. Следовательно, для обеспечения этих возможностей, были написаны триггеры удаления и обновления данных.

Интерфейс программы был выбран не многооконный, а многостраничный, что, по мнению автора, существенно упрощает и программирование, и доступ к данным со стороны пользователя.

Результатом курсовой работы стала база данных и приложение, корректно обрабатывающее все заявленные запросы и функции, не вызывающее ошибок времени исполнения.

Созданный материал пригоден для учебных целей, однако, для внедрения базы данных в существующий музей, могут понадобиться значительные изменения. Несмотря на этот факт, все цели и задачи, поставленные перед автором работы, были полностью реализованы.

**Список литературы**

1. Архангельский, А.Я. Программирование в С++ Builder 6. / А.Я. Архангельский –М.: Бином, 2002. – 1152 с.
2. Вендров, А.М. CASE-технологии. Современные методы и средства проектирования информационных систем. / А.М. Вендров. – М.: Финансы и статистика, 1998. -176 с.: ил.
3. Дейт, К. Введение в системы баз данных. / К. Дейт М., –СПб., Киев: Издательский дом «Вильямс». . –950 с.: ил.
4. Мамаев, Е. Microsoft SQL Server. / Е. Мамаев –СПб.:БХВ-Петербург, 2002. –1200 с.
5. Мюллер, Р. Базы данных и UML проектирование. / Р. Мюллер –М.: Лори, 2002. –420 с.
6. Нейбург, Э. Проектирование баз данных с помощью UML. / Э. Нейбург, Роберт Максимчук –М.: Вильямс, 2002. –288 с.
7. Тиори, Т. Проектирование структур баз данных, т. 1. / Т. Тиори, Дж. Фрай –М.: Мир, 1985.
8. Тихомиров, Ю. Microsoft SQL Server. Разработка приложений. / Ю. Тихомиров –СПб.: БХВ-Петербург, .
9. Харрингтон, Д.Л. Проектирование реляционных баз данных. / Д.Л. Харрингтон –М.: Лори. .
10. Холигвэрт, Д. С++Builder 5: руководство разработчика (том 1) / Д. Холигвэрт, Баттерфилд Д. –М.: Вильямс, . –866 с.
11. Хомоненко, А.Д. Базы данных: учебник для высших учебных заведений. / А.Д. Хомоненко, М.Г.Мальцев, В.М. Цыганков –СПб: КОРОНА принт, .

**Приложение**

В результате проделанной работы была создана база данных и программа для управления ею. На прилагаемом к работе диске находятся следующие данные:

|  |  |  |
| --- | --- | --- |
| **Каталог** | **Файл** | **Описание** |
| DB/Script | createBD.sql | сценарий создания базы данных |
| DB/Script | procedures.sql | сценарий создания хранимых процедур для вставки данных |
| DB/Script | views.sql | сценарий создания представлений для выборки данных из связанных таблиц |
| DB/Bin | магазин.mdf, магазин.ldf | двоичные файлы базы данных магазин |
| Программа/Проект | ShopCommander.bpr | Проект приложения  «Shop Commander» |
| Записка | Записка.doc | Текст пояснительной записки |