МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«РЯЗАНСКИЙ ГОСУДАРСТВЕННЫЙ РАДИОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
ИМЕНИ В.Ф. УТКИНА»

Кафедра «Государственного, муниципального и корпоративного управления»

МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

«Современные технологии баз данных»

Направление подготовки
38.04.04 «Государственное и муниципальное управление»

Профиль – Информационные технологии в государственном и муниципальном управлении

ОПОП академической магистратуры
«Государственное и муниципальное управление»

Формы обучения – очная, очно-заочная, заочная

Рязань

1. КОНТРОЛЬНЫЕ ВОПРОСЫ ПО ДИСЦИПЛИНЕ

Введение в БД. Реляционная модель данных.

1. Определение ИС, БД, СУБД. Понятие архитектуры клиент-сервер.
2. Модели БД. Типы СУБД. Функции СУБД.
3. Структурная часть РМД. Основные понятия.
4. Целостная часть РМД. Основные понятия.
5. Операции, нарушающие ссылочную целостность.
6. Основные стратегии поддержания ссылочной целостности.
7. Дополнительные стратегии поддержания ссылочной целостности.
8. Правила Кодда.
9. Информационное правило.
10. Гарантированный доступ к данным.
11. Систематическая поддержка отсутствующих значений .
12. Возможность изменения представлений. .
13. Наличие высокоуровневых операций управления данными.
14. Физическая независимость данных.
15. Логическая независимость данных.
16. Независимость контроля целостности.
17. Независимость от расположения.
18. Согласование языковых уровней.

Реляционная алгебра.

19. Операции объединения, пересечения и вычитания. Особенности объединения, пересечения и вычитания отношений.
20. Операции переименования, объединения, пересечения и декартова произведения.
21. Операции выборки и проекции.
22. Операция эквивалентного соединения.
23. Внутреннее соединение в реляционной алгебре.
24. Внешние соединения в реляционной алгебре.
25. Операции проекции и деления.

Язык SQL.

26. Синтаксис оператора SELECT. Пример.
27. Основные разделы языка SQL. Примеры операторов по каждому разделу.
28. Предложение SELECT и FROM оператора SELECT. Выборка. Исключение строк-дубликатов. Построение вычисляемых полей. Пример.
29. Сравнение значений в предложении WHERE. Операции IN, BETWEEN. Пример.
30. Операции LIKE, IS NULL в предложении WHERE. Пример.
31. Использование агрегатных функций в предложениях оператора SELECT. Пример.
32. Группировка с помощью предложения GROUP BY оператора SELECT. Пример.
33. Использование предложений GROUP BY и HAVING оператора SELECT. Пример.
34. Сортировка в операторе SELECT. Использование предложения TOP. Пример.
35. Декартово произведение с помощью оператора SELECT. Соединение с помощью предложения WHERE. Пример.
36. Типы соединений в предложении FROM. Пример.
37. Оператор объединения UNION. Особенности объединения запросов. Пример.
38. Операторы вычитания EXCEPT и пересечения INTERSECT. Особенности вычитания и пересечения запросов. Пример.
39. Подзапросы. Классификация подзапросов. Пример простого скалярного подзапроса.
40. Простые табличные подзапросы. Пример.
41. Сложные табличные подзапросы. Пример.
42. Оператор INSERT INTO. Оператор UPDATE. Пример.
43. Оператор UPDATE. Оператор DELETE. Пример.

Реализация операций реляционной алгебры в языке SQL.

44. Операция объединения в реляционной алгебре и ее реализация в языке SQL.
45. Операция пересечения в реляционной алгебре и ее реализация в языке SQL.

46. Операция вычитания в реляционной алгебре и ее реализация в языке SQL.
47. Операция декартового произведения в реляционной алгебре и ее реализация в языке SQL.
48. Операция выборки в реляционной алгебре и ее реализация в языке SQL.
49. Операция проекции в реляционной алгебре и ее реализация в языке SQL.
50. Операция внутреннего соединения в реляционной алгебре и ее реализация в языке SQL.
51. Операция левого внешнего соединения в реляционной алгебре и ее реализация в языке SQL.
52. Операция правого внешнего соединения в реляционной алгебре и ее реализация в языке SQL.
53. Операция деления в реляционной алгебре и ее реализация в языке SQL.
54. Реализация операции вычитания с помощью подзапросов.
55. Реализация операции пересечения с помощью подзапросов.
56. Реализация операции деления с помощью подзапросов.

СУБД SQL Server.

57. Создание БД в СУБД SQL Server. Пример.
58. Основные объекты БД SQL Server. Определения.
59. Системные БД SQL Server. Файлы БД. Скрипт создания БД. Переключение между БД. Подключение и отключение БД.
60. Операторы определения и использования переменных в Transact-SQL. Табличные переменные. Пример.
61. Условный оператор и оператор цикла в Transact-SQL. Пример.
62. Оператор выбора (простая форма) в Transact-SQL. Пример.
63. Оператор выбора (поисковая форма) в Transact-SQL. Пример.
64. Системные типы данных в СУБД SQL Server.
65. Создание таблиц. Ограничения первичного и внешнего ключа. Пример.
66. Создание таблиц. Ограничение внешнего ключа. Стратегии поддержания ссылочной целостности. Пример.
67. Создание таблиц. Ограничения CHECK, DEFAULT, UNIQUE. Пример.
68. Создание таблиц. Ограничения CHECK, UNIQUE. Задание имен ограничений. Пример.
69. Создание таблиц. Ограничения CHECK, DEFAULT. Задание столбцов-счетчиков. Пример.
70. Способы изменения структуры таблицы. Пример.
71. Функции работы с датой/временем в СУБД SQL Server. Пример.
72. Способы задания первичного ключа. Пример.
73. Способы задания внешнего ключа. Пример.
74. Умолчания. Способы задания умолчаний. Пример.
75. Уникальные значения. Способы задания уникальных значений. Пример.
76. Ограничения на значения. Способы задания ограничений на значения. Пример.
77. Определяемые пользователем типы данных. Пример.
78. Представления. Модифицируемые представления. Пример. Немодифицируемые представления. Пример.
79. Особенности модификации данных через представления. Пример.
80. Хранимые процедуры. Параметры процедур. Оператор выполнения ХП. Пример.
81. Триггеры. Определение DML и DDL триггеров. Пример.
82. AFTER-триггеры. Пример.
83. INSTEAD OF-триггеры. Пример.

Проектирование моделей данных

1. Проектирование БД. Основные задачи проектирования БД.
2. Проектирование БД. Основные этапы проектирования БД.
3. Постановка задачи проектирования БД. Основные способы проектирования ER-моделей.
4. Избыточность данных. Аномалии обновления. Пример.
5. Нормализация отношений. Типы зависимостей между атрибутами. Пример.
6. Нормализация отношений. 1НФ, 2НФ. Пример.
7. Нормализация отношений. 3НФ. Пример.
8. Нормализация отношений. БКНФ. Пример.
9. Нормализация отношений. 4НФ. Пример.
10. Проектирование реляционных БД с помощью ER-метода. Основные понятия.
11. Этапы проектирования реляционных БД с помощью ER-метода.

12. Правила перехода от ER-диаграммы к предварительным отношениям для бинарных связей 1:1 и 1:N.
13. Правила перехода от ER-диаграммы к предварительным отношениям для связей 1:N и N:N.
14. Правила перехода от ER-диаграммы к предварительным отношениям для связей N:N.
15. Правило формирования предварительных отношений при наличии супертипа и подтипов сущностей. Формирование отношений при наличии дополнительных связей между отдельными экземплярами подтипа и супертипа.
16. Правило формирования предварительных отношений при наличии рекурсии.
17. Современные средства проектирования БД. Обзор существующих решений.

Проверка правильности результатов проектирования моделей данных

18. Избыточность данных.
19. Аномалии обновления.
20. Процесс нормализации отношений.
21. Типы зависимостей между атрибутами.
22. Нормализация отношений 1НФ
23. Приведение отношений к 2НФ.
24. Приведение отношений к 3НФ.
25. Приведение отношений к БКНФ.
26. Приведение отношений к 4НФ.
27. Приведение отношений к 5НФ.

Распределенные БД

28. Распределенные базы данных. Основные моменты. Пример архитектуры.
29. Методы поддержки распределенных данных. Фрагментация.
30. Методы поддержки распределенных данных. Репликация.
31. Репликация. Модели тиражирования.
32. Методы поддержки распределенных данных. Распределенные ограничения целостности.
33. Методы поддержки распределенных данных. Распределенные запросы.
34. Методы поддержки распределенных данных. Распределенные транзакции.
35. Свойства идеальной РБД. Примеры.

Реализация сложных связей в базах данных

36. Проблемы циклических связей в БД. Способы разрешения проблемы.
37. Реализация наследования в БД. Виды наследования.
38. Реализация обычного и взаимоисключающего наследования в БД. Проблемы добавления новых данных.
39. Реализация обычного и взаимоисключающего наследования в БД. Проблемы обновления данных.
40. Реализация обычного и взаимоисключающего наследования в БД. Проблемы удаления данных.
41. Реализация обычного и законченного наследования в БД.
42. Реализация взаимоисключающего законченного наследования в БД. Проблемы добавления новых данных.
43. Реализация взаимоисключающего законченного наследования в БД. Проблемы обновления данных.
44. Реализация взаимоисключающего законченного наследования в БД. Проблемы удаления данных.
45. Проблемы рекурсивных связей в однокорневом дереве.
46. Проблемы добавления данных для рекурсивных связей в однокорневом дереве.
47. Проблемы обновления данных для рекурсивных связей в однокорневом дереве.
48. Проблемы удаления данных для рекурсивных связей в однокорневом дереве.

2. МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ

Выполнение лабораторных работ осуществляется в соответствии с методической литературой:

1. Громов А.Ю. Современные технологии разработки интегрированных информационных систем: учеб. пособие / А.Ю. Громов, Н.Н. Гринченко, Н.В. Шемонаев; Рязан. гос. радиотехн. ун-т. - Рязань, 2015. - 48 с. https://elib.rsreu.ru/ebs/download/562
2. Клиент-серверные приложения баз данных: учеб. пособие / А.В. Благодаров, Н.Н. Гринченко, А.Ю. Громов; Рязан. гос. радиотехн. ун-т. Рязань, 2017. 72 с. https://elib.rsreu.ru/ebs/download/2356
3. Гринченко Н.Н. Инструментальные средства поддержки проектирования баз данных: учеб. пособие / Н.Н. Гринченко, А.Ю. Громов; Рязан. гос. радиотехн. ун-т. - Рязань, 2015. - 48 с. https://elib.rsreu.ru/ebs/download/731

3. МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО НАПИСАНИЮ КУРСОВОЙ РАБОТЫ ПО ДИСЦИПЛИНЕ:

Целью курсовой работы является:

получение навыков проектирования моделей данных для разных предметных областей в различных нотациях, проверки правильности проектирования моделей данных, программирования запросов на диалектах языка SQL, создания и сопровождения баз данных в современных СУБД.

Задание на курсовую работу:

В соответствии с вариантом задания (см. таблицу) провести проектирование базы данных для указанной предметной области в заданной нотации. Разработать сценарий создания базы данных и основных объектов базы данных на заданном языке программирования БД, реализовать полученную структуру в заданной СУБД, создать основные объекты БД, требуемые пользователю.

№ варианта	Предметная область	Нотация для проектирования ER-модели	Диалект языка SQL	СУБД
1	Телекоммуникационный центр	П. Чена,	Transact-SQL	Microsoft SQL Server
2	Прокат спортинвентаря	Дж. Мартина,	PL/pgSQL	PostgreSQL
3	Выставка-продажа сельскохозяйственной продукции	Ч. Баркера,	PL/SQL	Oracle Database
4	Дневной стационар	Ж.-Р. Абриаля,	Transact-SQL	Microsoft SQL Server
5	Бюро трудоустройства	IDEF1X	PL/pgSQL	PostgreSQL
6	Показ-продажа моделей высокой моды	П. Чена,	PL/SQL	Oracle Database
7	Видеопрокат	Дж. Мартина,	Transact-SQL	Microsoft SQL Server
8	Чемпионат мира по футболу	Ч. Баркера,	PL/pgSQL	PostgreSQL
9	Риэлторская фирма	Ж.-Р. Абриаля,	PL/SQL	Oracle Database
10	Расписание занятий	IDEF1X	Transact-SQL	Microsoft SQL Server
11	Обувная мастерская	П. Чена,	Transact-SQL	Microsoft SQL Server
12	Тренажерный зал	Дж. Мартина,	PL/pgSQL	PostgreSQL
13	Салон красоты	Ч. Баркера,	PL/SQL	Oracle Database
14	Автовокзал	Ж.-Р. Абриаля,	Transact-SQL	Microsoft SQL Server
15	Туристическая фирма	IDEF1X	PL/pgSQL	PostgreSQL
16	Продажа компьютерной техники	П. Чена,	PL/SQL	Oracle Database
17	Автомагазин	Дж. Мартина,	Transact-SQL	Microsoft SQL

				Server
18	Спортивные товары	Ч. Баркера,	PL/pgSQL	PostgreSQL
19	Аэропорт	Ж.-Р. Абриаля,	PL/SQL	Oracle Database
20	Ресторан	IDEF1X	Transact-SQL	Microsoft SQL Server

Общие требования к работе:

В БД должно быть не менее 10 таблиц. Главные таблицы должны содержать не менее 10 строк, подчиненные — не менее 20.

Предполагается, что разрабатываемая БД будет представлять собой серверную часть информационной системы с клиент-серверной архитектурой, использующей двухзвенную модель DBS (сервер баз данных).

Провести инфологическое и даталогическогое проектирование БД. Проверить все таблицы БД на соответствие нормальной форме Бойса-Кодда. Предусмотреть контроль целостности данных на уровне ограничений, триггеров, правил, умолчаний и хранимых процедур. Все действия по внесению изменений в БД оформить в виде хранимых процедур.

При выполнении курсовой работы необходимо использовать материал лекций и рекомендуемую учебно-методическую литературу.

Требования к содержанию пояснительной записки:

Введение.

1. Постановка задачи и технико-экономическое обоснование разработки.
   - 1.1. Анализ предметной области, выявление необходимой пользователю функциональности.
   - 1.2. Обзор существующих аналогов.
   - 1.3. Выводы об актуальности разработки.
2. Разработка общей архитектуры.
3. Разработка моделей данных.
   - 3.1. Разработка инфологической модели данных.
     - 3.1.1. Выявление сущностей и связей.
     - 3.1.2. Построение ER-диаграммы.
   - 3.2. Разработка даталогической модели данных.
     - 3.2.1. Переход от ER-диаграммы к предварительным отношениям.
     - 3.2.2. Заполнение предварительных отношений атрибутами.
     - 3.2.3. Проверка предварительных отношений на соответствие нормальным формам.
     - 3.2.4. Построение схемы данных.
4. Разработка основных объектов БД.
   - 4.1. Задание частных ограничений целостности данных.
   - 4.2. Разработка представлений.
   - 4.3. Разработка запросов к БД.
   - 4.4. Разработка ХП, триггеров.

Заключение.

Список используемой литературы.

Приложение: SQL-скрипт для создания БД (должны быть включены все объекты БД – таблицы, представления, ХП и т.п.).

К защите представить:

- работоспособную БД, установленную на компьютере в кафедральной аудитории ;
- пояснительную записку в распечатанном виде;
- электронный каталог, содержащий пояснительную записку, файлы БД, сценарий создания основных объектов БД, сценарий заполнения таблиц.

В ходе выполнения курсовой работы рекомендуется придерживаться календарного плана, приведённого в таблице.

Содержание этапа	Продолжительность этапа
1. Выбор темы и утверждение технического задания.	2 недели
2. Постановка задачи и технико-экономическое	2 недели

обоснование разработки.	
3. Разработка общей архитектуры.	1 неделя
4. Разработка моделей данных.	3 недели
5. Создание схемы базы данных и заполнение таблиц данными.	1 неделя
6. Создание ограничений, представлений, хранимых процедур, триггеров.	2 недели
7. Оформление пояснительной записки.	1 неделя
8. Сдача курсовой работы на проверку.	1 неделя
9. Защита курсовой работы.	1 неделя

По результатам второго, четвертого и шестого этапов проводится промежуточный просмотр курсового проекта.

Типовые вопросы при защите курсовой работы:

1. Основные возможности современных СУБД.
2. Преимущества и недостатки работы в СУБД Microsoft SQL Server.
3. Преимущества и недостатки работы в СУБД Oracle Database.
4. Преимущества и недостатки работы в СУБД PostgreSQL.
5. Основные возможности диалекта Transact-SQL от СУБД Microsoft SQL Server.
6. Процедурное расширение языка PL/SQL в СУБД Oracle Database.
7. Основные возможности диалекта PL/pgSQL в СУБД PostgreSQL.
8. Анализ современных СУБД для решения типовых задач баз данных.
9. Особенности нотации для проектирования ER-моделей.
10. Особенности нотации П. Чена.
11. Особенности нотации Дж. Мартина.
12. Особенности нотации Ч. Баркера.
13. Особенности нотации Ж.-Р. Абриаля.
14. Особенности нотации IDEF1X.

3. ПРИМЕР ОФОРМЛЕНИЯ НЕКОТОРЫХ РАЗДЕЛОВ ПОЯСНИТЕЛЬНОЙ ЗАПИСКИ К КУРСОВОЙ РАБОТЕ

В качестве примера рассматривается предметная область «Предприятие по экспорту, продаже и сервисному обслуживанию российских автомобилей в странах Европы».

Перед проектированием моделей данных требуется провести тщательный анализ заданной предметной области, выявить функции, которые необходимы конечному пользователю системы.

3.1. Анализ предметной области, выявление необходимой пользователю функциональности

Предметная область – закупка автомобилей у российских автозаводов, экспорт, продажа и сервисное обслуживание автомобилей, проданных за границей.

В базе данных требуется хранить следующую *информацию*:

- данные о заводах-изготовителях: название (уникально), адрес, телефон;
- данные о представительствах фирмы – центральный офис (располагается в России), автосалоны (за рубежом) (автосалоны обеспечивают продажу и сервисное обслуживание автомобилей): тип (офис/автосалон), адрес (уникален), телефон;
- данные о сотрудниках фирмы: ФИО, телефон, должность, место работы;
- данные о клиентах фирмы – покупателях автомобилей: ФИО, телефон;
- данные об автомобилях, закупленных фирмой: IDАвтомобиля (уникальный числовой код), завод-изготовитель, модель, комплектация;

- данные о комплектациях автомобилей, предоставляемых фирмой: название комплектации (уникально, например Minimum, Medium, Maximum), цвет, объём двигателя, КПП, электропакет, число подушек безопасности, музыкальное оборудование;
- данные о закупках автомобилей у заводов-изготовителей (в России): IDАвтомобиля (уникально), ФИО сотрудника (осуществившего закупку), дата закупки, стоимость закупки;
- данные об экспорте автомобилей за границу: IDАвтомобиля (уникально), дата отправки, сотрудник (осуществивший отправку), автосалон (в который направлен автомобиль);
- данные о продажах автомобилей: IDАвтомобиля (уникально), сотрудник (продавец), клиент, дата продажи, стоимость продажи;
- данные о сервисном обслуживании автомобилей: IDАвтомобиля, сотрудник, дата обслуживания, причина обращения.

Анализ ограничений, накладываемых на модель

При этом на предметную область задачи, а следовательно, и на разрабатываемую базу данных накладываются следующие ограничения, или *бизнес-правила*:

- сотрудник может работать только в одном офисе/автосалоне;
- сотрудник может занимать только одну должность;
- автомобиль изготовлен лишь одним заводом, но каждый завод производит множество автомобилей;
- автомобиль может иметь только одну комплектацию, но у различных автомобилей может быть одинаковая комплектация;
- автомобиль может быть закуплен/экспортирован/продан лишь одним сотрудником, но каждый сотрудник может закупить/ экспортировать/продать/ множество автомобилей;
- сотрудник может обслужить несколько автомобилей, равно как и автомобиль может быть обслужен несколькими сотрудниками (поочерёдно);
- закупку и экспорт одного и того же автомобиля могут производить разные сотрудники;
- закупку и экспорт автомобилей могут производить только сотрудники центрального офиса с должностью «Менеджер», продажи автомобилей могут производить сотрудники представительств только с должностью «Менеджер», сервисное обслуживание могут производить только сотрудники с должностью «Автомеханик»;
- автомобиль может быть закуплен, экспортирован и продан лишь единожды, но обслуживание может как производиться многократно в одном или различных центрах, так и вообще не производиться;
- автомобиль может принадлежать лишь одному клиенту, но клиент может иметь несколько автомобилей;
- автомобиль не может быть экспортирован раньше, чем закуплен; продан раньше, чем экспортирован; обслужен раньше, чем продан;
- автомобиль не может быть экспортирован в центральный офис (так как он находится в России);
- автомобиль может быть продан только сотрудником того офиса, в который он был экспортирован.

3.2. Проектирование моделей данных

Процесс проектирования моделей данных должен включать следующие этапы:

1. Выявление сущностей и связей.
2. Построение ER – диаграммы.
3. Формирование предварительных отношений.
4. Подготовка списка атрибутов, которые были выделены в предметной области.
5. Распределение атрибутов по предварительным отношениям.
6. Проверка отношений на соответствие БКНФ.
7. Пересмотр ER-диаграммы.
8. Построение схемы базы данных.

Рассмотрим примеры оформления результатов каждого из этапов проектирования моделей данных.

3.2.1. Выявление сущностей и связей

В предметной области можно выделить следующие *сущности*:
1) Завод-изготовитель;
2) Офис;
3) Сотрудник;
4) Клиент;
5) Автомобиль;
6) Комплектация.

В предметной области можно выделить следующие *связи* между сущностями:
1. Сотрудник работает в офисе;
2. Сотрудник закупает автомобиль некоторой комплектации у завода-изготовителя;
3. Сотрудник экспортирует автомобиль в салон;
4. Сотрудник продаёт автомобиль клиенту;
5. Сотрудник обслуживает автомобиль.

3.2.2. Построение ER–диаграмм

На рисунках отображены степени связей между сущностями и классы принадлежности. Рассмотрим их подробнее.

1. Сотрудник работает в офисе:

Рисунок 1 – ER-диаграмма связи «сотрудник работает в офисе»

Тип связи: один ко многим. Сотрудник работает только в одном офисе, в офисе работает множество сотрудников.

Класс принадлежности: обязательный со стороны сотрудника (сотрудник обязательно работает в офисе) и необязательный со стороны офиса (в офисе не обязательно работают сотрудники).

2. Сотрудник закупает автомобиль некоторой комплектации у завода-изготовителя:

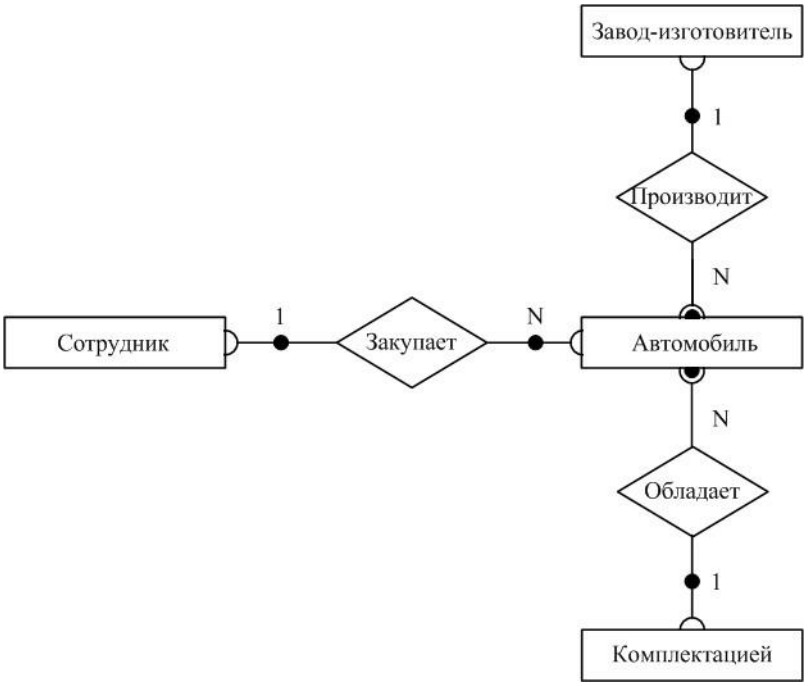

Рисунок 2 – ER-диаграмма связи
«сотрудник закупает автомобиль некоторой комплектации у завода-изготовителя»

Типы связей: один ко многим (сотрудник закупает множество автомобилей, автомобиль закуплен одним сотрудником; автомобиль обладает одной комплектацией, множество автомобилей могут обладать одинаковой комплектацией; завод производит множество автомобилей, автомобиль произведён одним заводом).

Классы принадлежности: в связи «сотрудник закупает автомобиль» – необязательные (сотрудник может не закупать автомобилей, автомобиль может быть не закуплен), в связях «завод производит автомобиль» и «автомобиль обладает комплектацией» - обязательные со стороны автомобиля (автомобиль обязательно произведён заводом, автомобиль обязательно обладает комплектацией).

3. Сотрудник экспортирует автомобиль в салон:

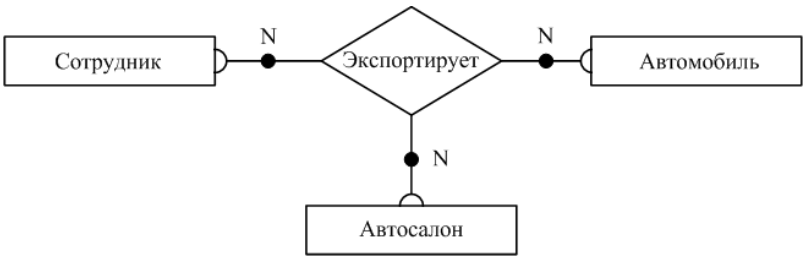

Рисунок 3 – ER-диаграмма связи «Сотрудник экспортирует автомобиль в салон»

Типы связей: многие ко многим (сотрудник экспортирует множество автомобилей во множество салонов, в салон экспортируется множество автомобилей множеством сотрудников).

Классы принадлежности: необязательные (сотрудник может не экспортировать автомобилей, в салон могут не экспортироваться автомобили, автомобиль может быть не экспортирован).

4. Сотрудник продаёт автомобиль клиенту:

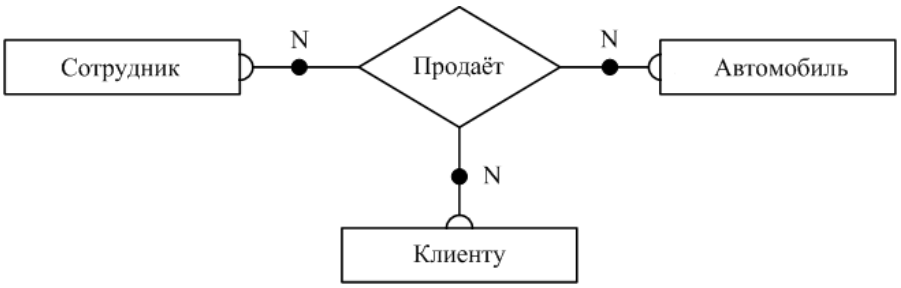

Рисунок 4 – ER-диаграмма связи «Сотрудник продаёт автомобиль клиенту»

Типы связей: многие ко многим (сотрудник продаёт множество автомобилей множеству клиентов, клиент может купить много автомобилей у различных сотрудников).

Классы принадлежности: необязательные (сотрудник может не продавать автомобилей, автомобиль может быть не продан).

5. Сотрудник обслуживает автомобиль:

Рисунок 5 – ER-диаграмма связи «Сотрудник обслуживает автомобиль»

Типы связей: многие ко многим (сотрудник обслуживает множество автомобилей, автомобиль может быть обслужен многими сотрудниками).

Классы принадлежности: необязательные (сотрудник может не обслуживать автомобилей, автомобиль может быть не обслужен).

Построение общей ER-диаграммы:

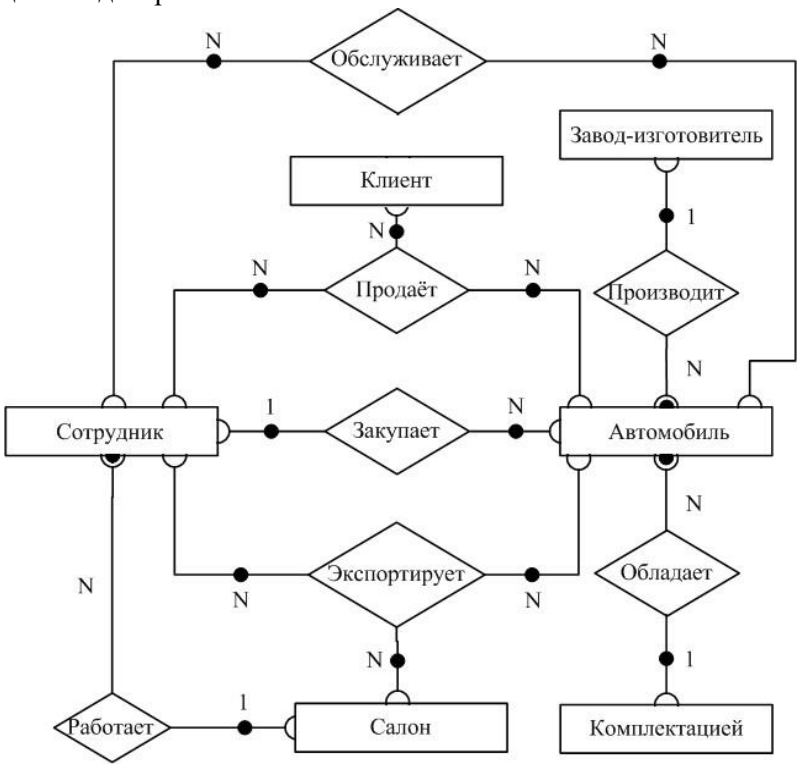

Рисунок 6 – Общая ER-диаграмма

По результатам первых двух этапов проектирования была получена инфологическая модель –

полная ER-диаграмма классов, представляющая собой объединение всех построенных ER-диаграмм для отдельных связей.

3.2.3. Формирование предварительных отношений по ER-диаграмме

Третий этап выполняется на основе специальных правил. При переходе от ER-диаграммы к предварительным отношениям используются следующие правила:

Правило 1: 1:1, класс принадлежности обеих сущностей – обязательный. Формируют 1 отношение, ключом является ключ любой из двух сущностей.

Правило 2: 1:1, класс принадлежности одной сущности – обязательный, второй – нет. Формируют 2 отношения по одному для сущности, ключом является ключ соответствующей сущности, при этом ключ сущности с необязательным классом принадлежности добавляется в отношение для сущностей с обязательным классом принадлежности.

Правило 3: 1:1, класс принадлежности обеих сущностей – необязательный. Формируют 3 отношения по одному для каждой сущности и одно для связи; ключом в первых двух отношениях является ключ соответствующей сущности. Отношение для связи должно содержать ключи обеих сущностей. В качестве ключа этого отношения можно принять ключ любой из сущностей.

Правило 4: 1:N, класс принадлежности связи со стороны N – обязательный. Формируется 2 отношения по одному для сущности, ключ – ключ сущности. В отношении со стороны N добавляется ключ односвязной сущности.

Правило 5: 1:N, класс принадлежности связи со стороны N – необязательный. Формируется 3 отношения: по одному для сущности и одно для связи. Ключи первых двух – ключи соответствующих сущностей, в отношении для связи содержатся ключи обеих сущностей, причём первичным ключом является ключ N-связной сущности.

Правило 6: N:N. Формируются три отношения: по одному для сущностей и одно для связи. Первичные ключи первых двух – ключи соответствующих сущностей. В отношении для связи должны содержаться ключи обеих сущностей, которые совместно являются первичным ключом отношения.

Правило 7: при наличии n-арной связи следует использовать n+1 отношение: по одному для каждой сущности (первичным ключом является ключ соответствующей сущности) и одно для связи (содержит ключи всех сущностей, которые совместно образуют первичный ключ).

По описанным выше ER-диаграммам и правилам получаем следующие предварительные отношения:

1. Сотрудник работает в офисе: по правилу 4 формируем 2 отношения: по одному для каждой сущности, где ключами являются ключи соответствующих сущностей:

Сотрудник(ФИОСотрудника, №Офиса),

Офис(№Офиса);

2. Сотрудник закупает автомобиль некоторой комплектации у завода-изготовителя:

для связи «Завод-изготовитель производит автомобиль» по правилу 4 формируем 2 отношения: по одному для каждой сущности, где ключами являются ключи соответствующих сущностей:

Завод-изготовитель(№Завода),

Автомобиль(IDАвтомобиля, №Завода);

для связи «Автомобиль обладает комплектацией» по правилу 4 формируем 2 отношения: по одному для каждой сущности, где ключами являются ключи соответствующих сущностей:

Комплектация(№Комплектации),

Автомобиль(IDАвтомобиля, №Завода, №Комплектации);

для связи «Сотрудник закупает автомобиль» по правилу 5 формируем 3 отношения: по одному для каждой сущности, где ключами являются ключи соответствующих сущностей и одно для связи, которое содержит ключи обеих сущностей и где ключом является ключ N-связи:

Сотрудник(ФИОСотрудника, №Офиса),

Автомобиль(IDАвтомобиля, №Завода, №Комплектации),

Закупка(IDАвтомобиля, ФИОСотрудника);

3. Сотрудник экспортирует автомобиль в салон: по правилу 7 формируем 4 отношения: по

одному для каждой сущности, где ключами являются ключи соответствующих сущностей и одно для связи, где ключ содержит ключи всех сущностей:

Сотрудник(ФИОСотрудника, №Офиса),
Офис(№Офиса),
Автомобиль(IDАвтомобиля, №Завода, №Комплектации),
Экспорт(IDАвтомобиля, ФИОСотрудника, №Офиса);

4. Сотрудник продаёт автомобиль клиенту: по правилу 7 формируем 4 отношения: по одному для каждой сущности, где ключами являются ключи соответствующих сущностей и одно для связи, где ключ содержит ключи всех сущностей:

Сотрудник(ФИОСотрудника, №Офиса),
Автомобиль(IDАвтомобиля,
№Завода, №Комплектации),
Клиент(ФИОКлиента),
Продажа(IDАвтомобиля, ФИОСотрудника, ФИОКлиента);

5. Сотрудник обслуживает автомобиль: по правилу 6 формируем 3 отношения: по одному для каждой сущности, где ключами являются ключи соответствующих сущностей и одно для связи, где ключ содержит ключи всех сущностей:

Сотрудник(ФИОСотрудника, №Офиса),
Автомобиль(IDАвтомобиля, №Завода, №Комплектации),
Обслуживание(IDАвтомобиля, ФИОСотрудника).

3.2.4. Подготовка списка атрибутов

На данном этапе подготавливаются атрибуты, которые должны храниться в базе данных. Это все, требуемые для конечного пользователя системы, атрибуты, кроме тех, которые уже были выделены на предыдущих этапах и попали схему БД. Совместим реализацию данного этапа со следующим (распределение атрибутов) по отношениям.

3.2.5. Распределение подготовленных атрибутов по предварительным отношениям

1) Офис(№Офиса, ТипОфиса, АдресОфиса, ТелефонОфиса);
2) Сотрудник(ФИОСотрудника, №Офиса, Должность, ТелефонСотрудника);
3) Завод-изготовитель(№Завода, НазваниеЗавода, АдресЗавода, ТелефонЗавода);
4) Комплектация(№Комплектации, НазваниеКомплектации, Цвет, ОбъёмДвигателя, КПП, Электропакет, ЧислоПодушекБезопасности, МузыкальноеОборудование);
5) Автомобиль(IDАвтомобиля, №Завода, Модель, №Комплектации);
6) Клиент(ФИОКлиента, ТелефонКлиента);
7) Закупка(IDАвтомобиля, ФИОСотрудника, ДатаЗакупки, СтоимостьЗакупки);
8) Экспорт(IDАвтомобиля, ФИОСотрудника, №Офиса, ДатаЭкспорта);
9) Продажа(IDАвтомобиля, ФИОСотрудника, ФИОКлиента, ДатаПродажи, СтоимостьПродажи);
10) Обслуживание(IDАвтомобиля, ФИОСотрудника, ДатаОбслуживания, ПричинаОбращения).

Как мы видим, поглощения отношений не произошло, так как каждое из них несёт дополнительную информацию по отношению к другим.

3.2.6. Проверка отношений на соответствие БКНФ

Существует несколько нормальных форм, каждой из которых соответствует определенный набор ограничений, и отношение находится в соответствующей нормальной форме, если оно удовлетворяет свойственному ей набору ограничений.

Выделяют 1НФ, 2НФ, 3НФ, БКНФ, 4НФ, 5НФ. Каждая нормальная форма сохраняет свойства предыдущих нормальных форм. Переход к более высокой НФ выполняется путем декомпозиции исходного отношения на два или более отношений, удовлетворяющих требованиям этой НФ.

Рассмотрим каждую из перечисленных нормальных форм более подробно.

1НФ. При 1НФ все атрибуты отношения должны быть простыми (атомарными, неделимыми) с точки зрения СУБД. Таблица не должна иметь повторяющихся записей, групп полей, необходимо наличие, по крайней мере, одного уникального индекса.

2НФ. Отношение находится в 2НФ, если оно находится в 1НФ и каждый неключевой атрибут функционально-полно зависит от первичного ключа. Из этого следует, что если ключ составной, то любой неключевой атрибут зависит от всего ключа и не зависит от какого-либо из входящих в него атрибутов.

3НФ. При третьей нормальной форме (3НФ) отношение должно находиться в 2НФ, а каждый неключевой атрибут нетранзитивно зависеть от ключа, т.е. отсутствуют функциональные зависимости между неключевыми атрибутами.

БКНФ. Нормальная форма Бойса-Кодда требует, чтобы в отношении были в 3НФ, и в нем отсутствовали зависимости атрибутов первичного ключа от неключевых атрибутов (ситуация, когда отношение имеет два или более возможных ключа, которые являются составными и имеют общий атрибут). В данном курсовом проекте стояло условие необходимости соответствия таблиц БД нормальной форме Бойса-Кодда.

Проверим наши отношения на принадлежность к НФ:

1) Офис:

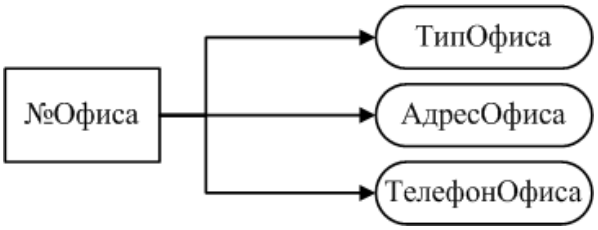

Рисунок 7 – Отношение «Офис»

2) Сотрудник:

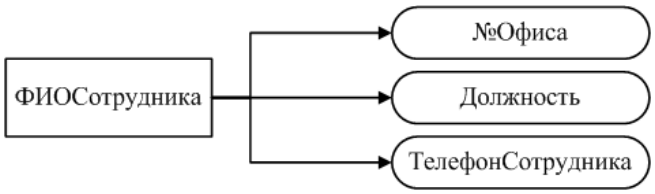

Рисунок 8 – Отношение «Сотрудник»

3) Завод-изготовитель:

Рисунок 9 – Отношение «Завод-изготовитель»

4) Комплектация:

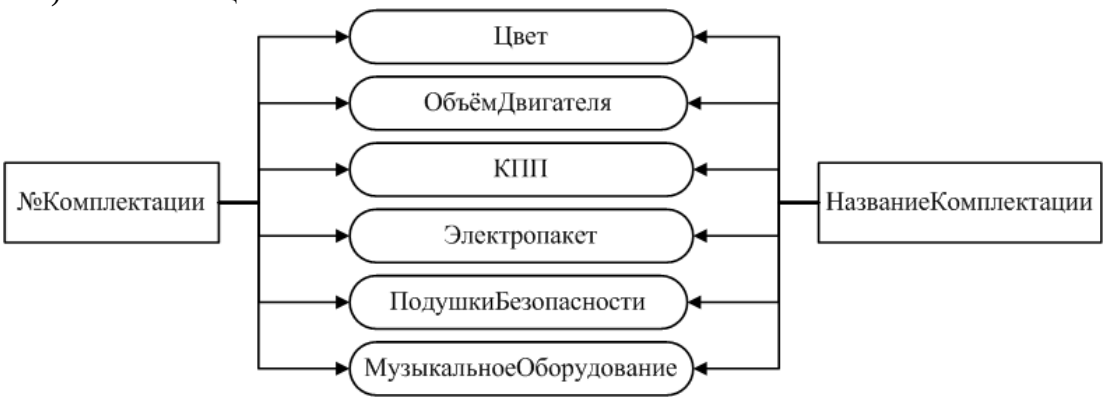

Рисунок 10 – Отношение «Комплектация»

5) Автомобиль:

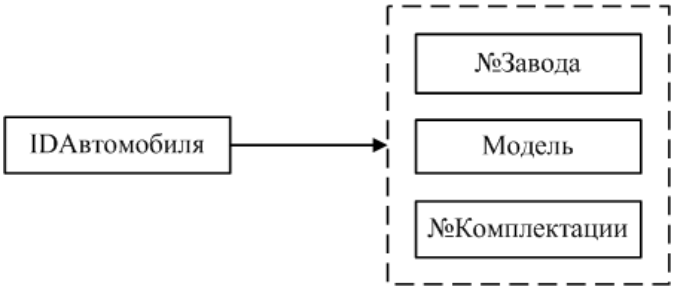

Рисунок 11 – Отношение «Автомобиль»

6) Клиент:

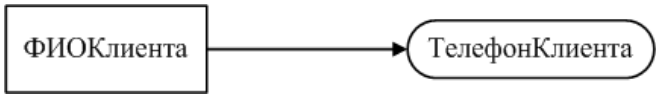

Рисунок 12 – Отношение «Клиент»

7) Закупка

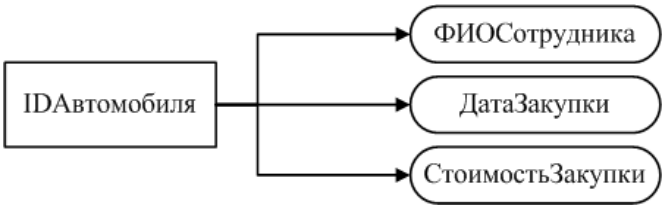

Рисунок 13 – Отношение «Закупка»

8) Экспорт:

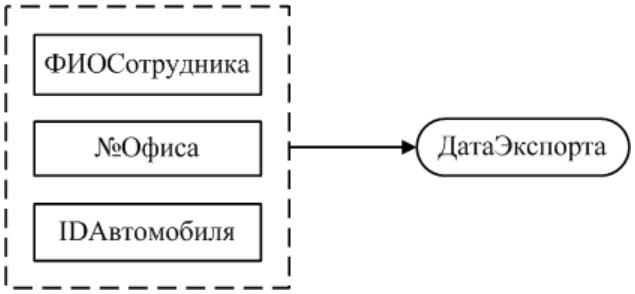

Рисунок 14 – Отношение «Экспорт»

9) Продажа:

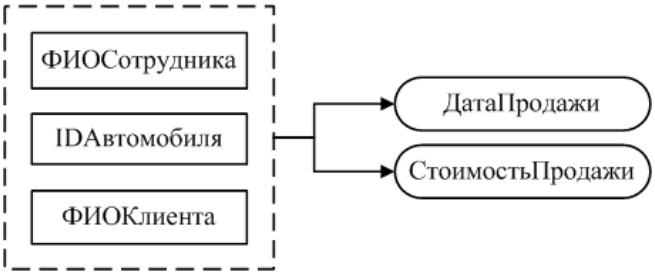

Рисунок 15 – Отношение «Продажа»

10) Обслуживание:

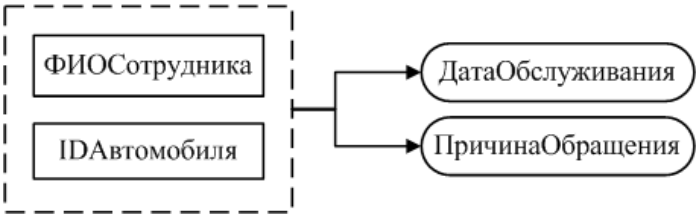

Рисунок 16 – Отношение «Обслуживание»

Все отношения находятся в 1НФ, так как на пересечении любой строки и столбца в отношениях находится ровно одно значение и так как все они имеют ключевые поля, которые по определению

являются уникальными индексами.

Все отношения находятся во 2НФ, так как они находятся в 1НФ и каждый неключевой атрибут функционально полно зависит от потенциального ключа.

Все отношения находятся во 3НФ, так как они находятся во 2НФ и в них нет транзитивных зависимостей неключевых атрибутов от потенциального ключа.

Все отношения находятся в БКНФ, так как они находятся в 3НФ и детерминанты всех функциональных зависимостей являются потенциальными ключами.

3.2.7. Пересмотр ER-диаграммы

Пересмотр ER-диаграммы не требуется, так как все отношения находятся в БКНФ и отсутствуют нераспределённые атрибуты.

3.2.8. Построение схемы БД

По результатам предыдущих семи этапов мы получили даталогическую модель нашей базы данных.

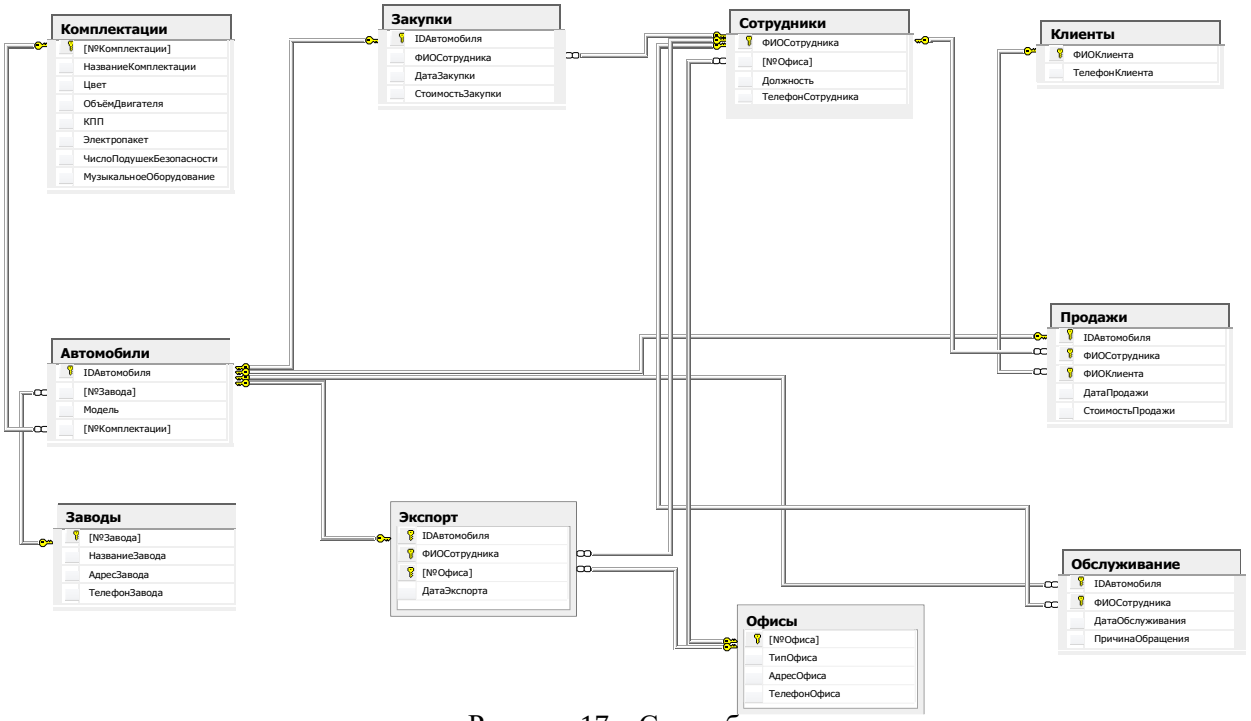

Рисунок 17 – Схема базы данных

3.3. Разработка БД и основных объектов БД.

3.3.1. Разработка БД и ограничений

Приведем пример создания базы данных Avto, которая будет включать ограничения первичных и внешних ключей, а также другие необходимые для пользователя ограничения:

```
create database Avto
     on
          (Name='Avto_Data',                  --имя файла
          Filename='C:\Avto_Data.mdf',        --путь к файлу
          Size=3,                             --начальный размер
          Maxsize=20,                         --максимальный размер
          Filegrowth=1)                  --шаг увеличения размера файла
     log on
          (Name='Avto_Log',
          Filename='C:\Avto_Log.ldf',
```

```
            Size=3,
            Maxsize=20,
            Filegrowth=1)
go
```

При создании таблиц в созданной базе данных описываются ограничения первичных и вторичных ключей для обеспечения ссылочной целостности базы данных.

Создание таблицы Офисы, содержащей следующую информацию: номер офиса (первичный ключ, столбец-счётчик), тип офиса (not null), адрес (not null), телефон:

```
create table Офисы([№Офиса] int identity(1, 1) primary key, ТипОфиса nvarchar(50) not null, АдресОфиса nvarchar(50) not null, ТелефонОфиса nvarchar(20))
```

Создание таблицы Сотрудники, содержащей следующую информацию: ФИО Сотрудника (первичный ключ), Номер офиса, в котором работает сотрудник (not null), должность (not null), телефон. Создаётся ограничение для связи поля Номер офиса с таблицей Офисы:

```
create table Сотрудники(ФИОСотрудника nvarchar(50) primary key, [№Офиса] int not null, Должность nvarchar(50) not null, ТелефонСотрудника nvarchar(20), constraint FK_Сотрудники_Офисы foreign key ([№Офиса]) references Офисы([№Офиса]))
```

Создание таблицы Заводы, содержащей следующую информацию: номер завода (первичный ключ, столбец-счётчик), название завода (not null и уникально), адрес завода, телефон:

```
create table Заводы([№Завода] int identity(1, 1) primary key, НазваниеЗавода nvarchar(10) not null unique, АдресЗавода nvarchar(50), ТелефонЗавода nvarchar(20))
```

Создание таблицы Комплектации, содержащей следующую информацию: Номер комплектации (первичный ключ, столбец-счётчик), название комплектации (not null и уникально), цвет (not null), объём двигателя (not null), коробка передач (not null), электропакет (not null), число подушек безопасности (not null), музыкальное оборудование (not null):

```
create table Комплектации([№Комплектации] int identity(1, 1) primary key, НазваниеКомплектации nvarchar(20) not null unique, Цвет nvarchar(15) not null, ОбъёмДвигателя real not null, КПП nvarchar(20) not null, Электропакет nvarchar(20) not null, ЧислоПодушекБезопасности int not null, МузыкальноеОборудование nvarchar(20) not null)
```

Создание таблицы Автомобили, содержащей помимо приведённой ниже информации ограничения для связи полей Номер завода и Номер комплектации с таблицами Заводы и Комплектации:

```
create table Автомобили(IDАвтомобиля int primary key, [№Завода] int not null, Модель nvarchar(20) not null, [№Комплектации] int not null
constraint FK_Автомобили_Заводы foreign key ([№Завода]) references Заводы([№Завода]),
        constraint FK_Автомобили_Комплектации foreign key ([№Комплектации]) references Комплектации([№Комплектации]))
```

Создание таблицы Клиенты:

```
create table Клиенты (ФИОКлиента nvarchar(50) primary key, ТелефонКлиента nvarchar(20))
```

Создание таблицы Закупки, содержащей ограничения для связи полей IDАвтомобиля, ФИО Сотрудника с базовыми таблицами:

```
create table Закупки(IDАвтомобиля int unique, ФИОСотрудника nvarchar(50), ДатаЗакупки datetime not null, СтоимостьЗакупки money not null,
constraint PK_Закупки primary key (IDАвтомобиля),
constraint FK_Закупки_Автомобили foreign key (IDАвтомобиля) references Автомобили(IDАвтомобиля),
constraint FK_Закупки_Сотрудники foreign key (ФИОСотрудника) references Сотрудники(ФИОСотрудника))
```

Создание таблицы Экспорт, содержащей ограничения для связи полей IDАвтомобиля, ФИО Сотрудника и Номер офиса с базовыми таблицами:

```
create table Экспорт(IDАвтомобиля int unique, ФИОСотрудника nvarchar(50), [№Офиса] int not null, ДатаЭкспорта datetime not null,
constraint PK_Экспорт primary key (ФИОСотрудника, [№Офиса], IDАвтомобиля),
constraint FK_Экспорт_Сотрудники foreign key (ФИОСотрудника) references Сотрудники(ФИОСотрудника),
constraint FK_Экспорт_Офисы foreign key ([№Офиса]) references Офисы([№Офиса]),
constraint FK_Экспорт_Автомобили foreign key (IDАвтомобиля) references Автомобили(IDАвтомобиля))
```

Создание таблицы Продажи, содержащей ограничения для связи полей IDАвтомобиля, ФИО Сотрудника и ФИО клиента с базовыми таблицами:

```
create table Продажи(IDАвтомобиля int unique, ФИОСотрудника nvarchar(50), ФИОКлиента nvarchar(50), ДатаПродажи datetime not null, СтоимостьПродажи money not null,
constraint PK_Продажи primary key (ФИОСотрудника, IDАвтомобиля, ФИОКлиента),
constraint FK_Продажи_Сотрудники foreign key (ФИОСотрудника) references Сотрудники(ФИОСотрудника),
constraint FK_Продажи_Автомобили foreign key (IDАвтомобиля) references Автомобили(IDАвтомобиля),
constraint FK_Продажи_Клиенты foreign key (ФИОКлиента) references Клиенты(ФИОКлиента))
```

Создание таблицы Обслуживание, содержащей ограничения для связи полей IDАвтомобиля, ФИО Сотрудника с базовыми таблицами:

```
create table Обслуживание(IDАвтомобиля int, ФИОСотрудника nvarchar(50), ДатаОбслуживания datetime not null, ПричинаОбращения nvarchar(50),
constraint PK_Обслуживание primary key (ФИОСотрудника, IDАвтомобиля),
constraint FK_Обслуживание_Сотрудники foreign key (ФИОСотрудника) references Сотрудники(ФИОСотрудника),
constraint FK_Обслуживание_Автомобили foreign key (IDАвтомобиля) references Автомобили(IDАвтомобиля))
```

Таким образом, в нашей базе данных содержится десять таблиц, что соответствует требованиям, предъявляемым к данной курсовой работе.

3.3.2. Разработка правил

Для ограничения вводимых пользователем значений было разработано правило «Больше нуля»:

```
create rule БольшеНуля as @x>0
```

Данное правило было установлено для тех полей таблиц базы данных, величина которых не может принимать отрицательные значения (это номер офиса, объём двигателя, число подушек безопасности, номер завода, номер комплектации, стоимость закупки и продажи):

```
exec sp_bindrule 'БольшеНуля', 'Сотрудники.№Офиса'
exec sp_bindrule 'БольшеНуля', 'Комплектации.ОбъёмДвигателя'
exec sp_bindrule 'БольшеНуля',
 'Комплектации.ЧислоПодушекБезопасности'
exec sp_bindrule 'БольшеНуля', 'Автомобили.№Завода'
exec sp_bindrule 'БольшеНуля', 'Автомобили.№Комплектации'
exec sp_bindrule 'БольшеНуля', 'Закупки.СтоимостьЗакупки'
exec sp_bindrule 'БольшеНуля', 'Экспорт.№Офиса'
exec sp_bindrule 'БольшеНуля', 'Продажи.СтоимостьПродажи'
```

Надобности в разработке большего количества правил нет.

3.3.3. Разработка умолчаний

Был разработан ряд умолчаний, в соответствии с которыми будут задаваться значения неопределённым величинам. Эти значения определялись по принципу наиболее популярного варианта значения поля.

1. По умолчанию тип офиса будет приниматься равным 'Торгово-сервисное представительство', поскольку центральный офис всего один:

```
create default ТипОфисаПоУмолч
as 'Торгово-сервисное представительство'
go
exec sp_bindefault 'ТипОфисаПоУмолч', 'Офисы.ТипОфиса'
go
```

2. Должность сотрудника по умолчанию принимается равной 'Менеджер', поскольку это преобладающая профессия работников предприятия

```
create default ДолжностьПоУмолч as 'Менеджер'
go
exec sp_bindefault 'ДолжностьПоУмолч', 'Сотрудники.Должность'
go
```

3. Цвет по умолчанию – Серебристый:

```
create default ЦветПоУмолч as 'Серебристый'
go
exec sp_bindefault 'ЦветПоУмолч', 'Комплектации.Цвет'
go
```

4. По умолчанию объём двигателя принимается равным 1.5 литра:

```
create default ОбъёмДвигателяПоУмолч as 1.5
go
exec sp_bindefault 'ОбъёмДвигателяПоУмолч',
 'Комплектации.ОбъёмДвигателя'
go
```

5. КПП по умолчанию – Механическая (как наиболее распространённая):

```
create default КППоУмолч as 'Механическая'
go
exec sp_bindefault 'КППоУмолч', 'Комплектации.КПП'
go
```

6. Электропакет по умолчанию Частичный:

```
create default ЭлектропакетПоУмолч as 'Частичный'
go
exec sp_bindefault 'ЭлектропакетПоУмолч',
'Комплектации.Электропакет'
go
```

7. По умолчанию число подушек безопаснсти равно 2:

```
create default ЧислоПодушекБезопасностиПоУмолч as 2
go
exec sp_bindefault 'ЧислоПодушекБезопасностиПоУмолч',
'Комплектации.ЧислоПодушекБезопасности'
go
```

8. По умолчанию устанавливается музыкальное оборудование фирмы Mystery

```
create default МузыкальноеОборудованиеПоУмолч as 'Mystery'
go
exec sp_bindefault 'МузыкальноеОборудованиеПоУмолч',
'Комплектации.МузыкальноеОборудование'
go
```

9. По умолчанию завод-изготовитель - ВАЗ (№1) так как основную массу закупаемых автомобилей составляет его продукция:

```
create default НомерЗаводаПоУмолч as 1
go
exec sp_bindefault 'НомерЗаводаПоУмолч', 'Автомобили.№Завода'
go
```

10. По умолчанию принимается комплектация Средняя (№2):

```
create default НомерКомплектацииПоУмолч as 2
go
exec sp_bindefault 'НомерКомплектацииПоУмолч',
'Автомобили.№Комплектации'
go
```

11. Стоимость автомобиля по умолчанию принимается равной 300000 р:

```
create default СтоимостьПоУмолч as 300000
go
exec sp_bindefault 'СтоимостьПоУмолч', 'Закупки.СтоимостьЗакупки'
exec sp_bindefault 'СтоимостьПоУмолч', 'Продажи.СтоимостьПродажи'
go
```

12. По умолчанию автомобили экспортируются во 2й офис (в Германию):

```
create default НомерОфисаПоУмолч as 2
go
exec sp_bindefault 'НомерОфисаПоУмолч', 'Экспорт.№Офиса'
exec sp_bindefault 'НомерОфисаПоУмолч', 'Сотрудники.№Офиса'
go
```

3.3.4. Разработка представлений

Представления в базе данных создавались с учётом их практической надобности. Некоторые из них использовались для облегчения процесса разработки БД.

1. Создание представления, позволяющего просмотреть список менеджеров:

```
create view Менеджеры
     as select ФИОСотрудника, [№Офиса], Должность from Сотрудники
where Должность='Менеджер'
```

2. Создание представления, позволяющего просмотреть список автомехаников:

```
create view Автомеханики
	as select ФИОСотрудника, [№Офиса], Должность from Сотрудники where Должность='Автомеханик'
```

3. Создание представления, позволяющего просмотреть список непроданных автомобилей:

```
create view НепроданныеАвтомобили
	as select * from Автомобили where IDАвтомобиля not in (select IDАвтомобиля from Продажи)
```

4. Создание представления, позволяющего просмотреть список автомобилей, которым не предоставлялось сервисное обслуживание:

```
create view НеобслуженныеАвтомобили
	as select * from Автомобили where (IDАвтомобиля in (select IDАвтомобиля from Продажи)) and (IDАвтомобиля not in (select IDАвтомобиля from Обслуживание))
```

5. Создание представления, позволяющего просмотреть прибыль от продажи каждого автомобиля:

```
create view ПрибыльОтПродажи
	as select Продажи.IDАвтомобиля, Продажи.ФИОСотрудника, Закупки.СтоимостьЗакупки, Продажи.СтоимостьПродажи,
		Прибыль=(СтоимостьПродажи-СтоимостьЗакупки) from (Закупки inner join Автомобили on Закупки.IDАвтомобиля=Автомобили.IDАвтомобиля inner join Продажи on Автомобили.IDАвтомобиля=Продажи.IDАвтомобиля)
```

6. Создание представления, позволяющего просмотреть прибыль, полученную каждым менеджером от продаж автомобилей:

```
create view ПрибыльПоМенеджерам
	as select distinct Менеджеры.ФИОСотрудника, ПрибыльСотрудника=(select sum(ПрибыльОтПродажи.Прибыль) from ПрибыльОтПродажи
		where Менеджеры.ФИОСотрудника=ПрибыльОтПродажи.ФИОСотрудника group by ФИОСотрудника)
		from Менеджеры left join ПрибыльОтПродажи on Менеджеры.ФИОСотрудника=ПрибыльОтПродажи.ФИОСотрудника
```

3.3.5. Разработка хранимых процедур

Разработанные хранимые процедуры можно подразделить на четыре основных типа: для добавления строк в базовые таблицы, для изменения данных базовых таблиц, для выборки данных и для удаления данных. Для каждой процедуры приведено по два запроса, первый из которых приводит к ошибке, а второй – верен.

1) Процедуры добавления данных в таблицы-справочники:

Добавление данных в таблицу Автомобили:

```
drop proc Добавить_Автомобиль
go
create proc Добавить_Автомобиль(@IDАвтомобиля int, @НомерЗавода int, @Модель nvarchar(20), @НомерКомплектации int)
	as if not exists(select * from Автомобили where IDАвтомобиля=@IDАвтомобиля)
```

```
            insert into Автомобили
            values(@IDАвтомобиля,       @НомерЗавода,       @Модель,
@НомерКомплектации)
        else print('Автомобиль с таким ID уже существует')
        go

    exec Добавить_Автомобиль 1, 1, '2114', 2
    exec Добавить_Автомобиль 21, 5, 'Мобиль', 3
```

Добавление данных в таблицу Заводы:

```
    drop proc Добавить_Завод
    go
    create   proc   Добавить_Завод(@НомерЗавода   int,   @НазваниеЗавода
nvarchar(10), @АдресЗавода nvarchar(50),
                                        @ТелефонЗавода nvarchar(20))
        as     if     not     exists(select     *     from     Заводы
where([№Завода]=@НомерЗавода) or (НазваниеЗавода=@НазваниеЗавода))
            insert into Заводы
            values(@НазваниеЗавода, @АдресЗавода, @ТелефонЗавода)
        else print('Информация о заводе с таким номером или названием
уже существует')
        go

    exec Добавить_Завод 1, 'ВАЗ', 'Тольятти, Автомобилистов, 1', '55-
66-77'
    exec Добавить_Завод 6, 'Урал', 'Екатеринбург, Полевая, 6', '543-21-
00'
```

Добавление данных в таблицу Клиенты:

```
    drop proc Добавить_Клиента
    go
    create     proc     Добавить_Клиента(@ФИОКлиента     nvarchar(50),
@ТелефонКлиента nvarchar(20))
        as   if   not   exists(select   *   from   Клиенты   where
ФИОКлиента=@ФИОКлиента)
            insert into Клиенты
            values (@ФИОКлиента,  @ТелефонКлиента)
        else print('Клиент с таким ФИО уже существует')
        go

    exec Добавить_Клиента 'Блюм Б. Б.', null
    exec Добавить_Клиента 'Стрейзанд С. С.', '89567884409'
```

Добавление данных в таблицу Комплектации:

```
    drop proc Добавить_Комплектацию
    go
    create    proc    Добавить_Комплектацию(@НомерКомплектации    int,
@НазваниеКомплектации nvarchar(20), @Цвет nvarchar(15), @ОбъёмДвигателя
real,      @КПП      nvarchar(20),      @Электропакет      nvarchar(20),
@ЧислоПодушекБезопасности int, @МузыкальноеОборудование nvarchar(20))
        as    if    not    exists(select    *    from    Комплектации
where(@НомерКомплектации=[№Комплектации]                                or
НазваниеКомплектации=@НазваниеКомплектации))
```

```
            insert into Комплектации
            values(@НазваниеКомплектации,  @Цвет,  @ОбъёмДвигателя,
@КПП, @Электропакет,
                        @ЧислоПодушекБезопасности,
@МузыкальноеОборудование)
        else print('Комплектация с таким номером или названием уже
существует')
        go

    exec  Добавить_Комплектацию  1,  'Абракадабра',  'Синий',  2,
'Механическая', 'Полный', 3, 'AIWA'
    exec  Добавить_Комплектацию  4,  'Люкс',  'Чёрный',  2.1,
'Автоматическая', 'Полный', 7, 'BeatsAudio'
```

Добавление данных в таблицу Офисы:

```
    create proc Добавить_Офис(@НомерОфиса int, @ТипОфиса nvarchar(50),
@АдресОфиса nvarchar(50), @ТелефонОфиса nvarchar(20))
        as  if  not  exists(select  *  from  Офисы  where
[№Офиса]=@НомерОфиса)
        insert into Офисы
        values(@ТипОфиса, @АдресОфиса, @ТелефонОфиса)
        else print('Офис с таким номером уже существует')
        go

    exec  Добавить_Офис  2,  'Торгово-сервисное  представительство',
'Германия, Мюнхен, Зеештрассе, 16', '444-55-66'
    exec  Добавить_Офис  8,  'Торгово-сервисное  представительство',
'Швеция, Копенгаген, Карла Маркса, 15', '777-55-66'
```

Добавление данных в таблицу Сотрудники:

```
    create  proc  Добавить_Сотрудника(@ФИОСотрудника  nvarchar(50),
@НомерОфиса int, @Должность nvarchar(50),
                                        @ТелефонСотрудника
nvarchar(20))
        as  if  not  exists(select  *  from  Сотрудники  where
ФИОСотрудника=@ФИОСотрудника)
        insert into Сотрудники
        values(@ФИОСотрудника,      @НомерОфиса,      @Должность,
@ТелефонСотрудника)
        else print('Сотрудник с таким ФИО уже существует')
        go

    exec  Добавить_Сотрудника  'Абрамов  А.  А.',  5,  'Автомеханик',
    NULL
    exec  Добавить_Сотрудника  'Абрамович  А.  А.',  7,  'Автомеханик',
    '89038372639'
```

2). Процедуры модификации данных в таблицах-справочниках:

Изменение требований к комплектациям автомобилей:
Изменение количества подушек безопасности:

```
    create proc Изменить_Число_Подушек(@Штук int)
```

```
          as              update              Комплектации              set
ЧислоПодушекБезопасности=ЧислоПодушекБезопасности+@Штук
          go

     exec Изменить_Число_Подушек 1
     exec Изменить_Число_Подушек -1
```

Изменение объёма двигателя:

```
     create proc Изменить_Объём_Двигателя(@Литров real)
          as              update              Комплектации              set
ОбъёмДвигателя=ОбъёмДвигателя+@Литров
          go

     exec Изменить_Объём_Двигателя 1
     exec Изменить_Объём_Двигателя -1
```

Данный тип процедур нерационален в представленной базе данных, так как она имеет дело с уже произведёнными автомобилями, характеристики которых изменить уже невозможно. Данные запросы разобраны для теоретического примера.

3) Процедуры выборки данных

Определение фамилии менеджера, закупившего автомобиль по IDАвтомобиля:

```
     create proc Закупивший_Сотрудник(@IDАвтомобиля int)
          as
          declare @ФИОСотрудника nvarchar(50)
          select @ФИОСотрудника = ФИОСотрудника from Закупки where
Закупки.IDАвтомобиля=@IDАвтомобиля
          print    'Автомобиль    с    ID    равным    '+convert(char(2),
@IDАвтомобиля)+' закупил '+@ФИОСотрудника
     go

     exec Закупивший_Сотрудник 12
```

Слежение за движением автомобиля:

```
     create proc Где_Автомобиль(@IDАвтомобиля int)
          as
          if    not    exists    (select    *    from    Закупки    where
Закупки.IDАвтомобиля=@IDАвтомобиля)
          print    'Автомобиль    с    ID    равным    '+convert(char(2),
@IDАвтомобиля)+' не был закуплен'
          else
               begin
               if   not   exists   (select   *   from   Продажи   where
Продажи.IDАвтомобиля=@IDАвтомобиля)
               print   'Автомобиль   с   ID   равным   '+convert(char(2),
@IDАвтомобиля)+' был закуплен, но не был продан'
               else
                    begin
                    if not exists (select * from Обслуживание where
Обслуживание.IDАвтомобиля=@IDАвтомобиля)
                    print 'Автомобиль с ID равным '+convert(char(2),
@IDАвтомобиля)+' был закуплен, продан, но не был обслужен'
```

```
                    else
                         print     'Автомобиль     с     ID     равным
'+convert(char(2), @IDАвтомобиля)+' был закуплен, продан и был обслужен'
                    end
               end

     exec Где_Автомобиль 20
```

Расчёт прибыли фирмы в месяце, соответствующем введённой дате:

```
     drop proc Месячная_Прибыль
     create proc Месячная_Прибыль(@Дата datetime)
          as
          declare @Прибыль money
          select @Прибыль=0
          select   @Прибыль=sum(СтоимостьПродажи-СтоимостьЗакупки)   from
Закупки inner join Продажи
               on    Закупки.IDАвтомобиля=Продажи.IDАвтомобиля    where
month(ДатаПродажи)=month(@Дата) and
               year(ДатаПродажи)=year(@Дата)
          if @Прибыль > 0
          print    'В    '+convert(char(2),    month(@Дата))+'-м    месяце
'+convert(char(4), year(@Дата))+'-го года прибыль составила '
                         +convert(char(10), @Прибыль)+' рублей'
          else print 'В '+convert(char(2), month(@Дата))+'-м месяце
'+convert(char(4), year(@Дата))+'-го года прибыли не было'

     exec Месячная_Прибыль '12.11.11'
```

4) Процедуры удаления данных из таблиц-справочников:

Удаление сотрудника по его ФИО:

```
     create proc Удалить_Сотрудника(@ФИОСотрудника nvarchar(50))
          as
          delete from Сотрудники where ФИОСотрудника=@ФИОСотрудника
          go
```

Удаление завода по его номеру:

```
     create proc Удалить_Завод(@НомерЗавода int)
          as
          delete from Заводы where [№Завода]=@НомерЗавода
          go
```

Удаление клиента по его ФИО:

```
     create proc Удалить_Клиента(@ФИОКлиента nvarchar(50))
          as
          delete from Клиенты where ФИОКлиента=@ФИОКлиента
          go
```

Удаление комплектации по её номеру:

```
create proc Удалить_Комплектацию(@НомерКомплектации int)
     as
     delete from Комплектации where
[№Комплектации]=@НомерКомплектации
     go
```

Удаление офиса по его номеру:

```
create proc Удалить_Офис(@НомерОфиса int)
     as
     delete from Офисы where [№Офиса]=@НомерОфиса
     go
```

Удаление автомобиля по его ID:

```
create proc Удалить_Автомобиль(@IDАвтомобиля int)
     as
     delete from Автомобили where IDАвтомобиля=@IDАвтомобиля
     go
```

3.3.6. Разработка триггеров

Для контроля за соблюдением бизнес-правил, описанных ранее, к таблицам базы данных были подкреплены следующие триггеры:

1. Создание триггера, контролирующего то, чтобы закупки автомобилей осуществлялись только сотрудниками центрального офиса, имеющими должность Менеджер:

```
create trigger Контроль_Закупок on Закупки
     instead of insert, update
     as
     declare @IDАвтомобиля int, @ФИОСотрудника nvarchar(50),
@ДатаЗакупки datetime, @СтоимостьЗакупки money
     select @IDАвтомобиля=IDАвтомобиля,
@ФИОСотрудника=ФИОСотрудника, @ДатаЗакупки=ДатаЗакупки,
          @СтоимостьЗакупки=СтоимостьЗакупки from inserted
     if (@ФИОСотрудника in (select ФИОСотрудника from Сотрудники
where [№Офиса]=1 and Должность='Менеджер'))
          and @IDАвтомобиля in (select IDАвтомобиля from
Автомобили)
     insert into Закупки values (@IDАвтомобиля, @ФИОСотрудника,
@ДатаЗакупки, @СтоимостьЗакупки)
     else print 'Закупка невозможна'

insert into Закупки values (21, 'Сидоров С. С.', '11.11.2011',
400000)
```

2. Создание триггера, контролирующего то, чтобы дата экспорта была позже даты закупки, чтобы экспорт производился только сотрудниками центрального офиса, имеющими должность Менеджер и чтобы автомобиль не мог быть экспортирован в центральный офис:

```
create trigger Контроль_Экспорта on Экспорт
     instead of insert, update
     as
     declare @IDАвтомобиля int, @ФИОСотрудника nvarchar(50),
@НомерОфиса int, @ДатаЭкспорта datetime, @ДатаЗакупки datetime
     select @IDАвтомобиля=IDАвтомобиля,
```

```
@ФИОСотрудника=ФИОСотрудника, @НомерОфиса=[№Офиса],
                    @ДатаЭкспорта=ДатаЭкспорта from inserted
          select    @ДатаЗакупки=ДатаЗакупки    from    Закупки    where
Закупки.IDАвтомобиля=@IDАвтомобиля
          if (@IDАвтомобиля in (select IDАвтомобиля from Закупки)) and
(@ДатаЭкспорта >= @ДатаЗакупки) and
               (@ФИОСотрудника in (select ФИОСотрудника from Сотрудники
where [№Офиса]=1 and Должность='Менеджер')) and
               (@НомерОфиса <>1)
          insert  into  Экспорт  values  (@IDАвтомобиля,  @ФИОСотрудника,
@НомерОфиса, @ДатаЭкспорта)
          else print 'Экспорт невозможен'
     go

     insert into Экспорт values (20, 'Сидоров С. С.', 7, '11.09.11')
     insert into Экспорт values (20, 'Кузьмин К. К.', 7, '11.11.11')
     insert into Экспорт values (20, 'Сидоров С. С.', 1, '11.11.11')
     insert into Экспорт values (20, 'Сидоров С. С.', 7, '11.11.11')
```

3. Создание триггера, контролирующего то, чтобы дата экспорта была позже даты закупки и чтобы автомобиль был продан в том офисе, куда был экспортирован только сотрудником, имеющим должность Менеджер:

```
     create trigger Контроль_Продаж on Продажи
          instead of insert, update
          as
          declare   @IDАвтомобиля   int,   @ФИОСотрудника   nvarchar(50),
@ФИОКлиента nvarchar(50), @ДатаПродажи datetime,
                    @СтоимостьПродажи money, @ДатаЭкспорта datetime
          select                                  @IDАвтомобиля=IDАвтомобиля,
@ФИОСотрудника=ФИОСотрудника, @ФИОКлиента=ФИОКлиента,
                    @ДатаПродажи=ДатаПродажи,
@СтоимостьПродажи=СтоимостьПродажи from inserted
          select    @ДатаЭкспорта=ДатаЭкспорта    from    Экспорт    where
Экспорт.IDАвтомобиля=@IDАвтомобиля
          if (@IDАвтомобиля in (select IDАвтомобиля from Экспорт)) and
(@ДатаПродажи >= @ДатаЭкспорта) and
               (@ФИОСотрудника in (select ФИОСотрудника from Сотрудники
where Сотрудники.[№Офиса] in
                    (select     [№Офиса]     from     Экспорт     where
IDАвтомобиля=@IDАвтомобиля)) and
                    (@ФИОСотрудника   in   (select   ФИОСотрудника   from
Сотрудники where [№Офиса]<>1 and Должность='Менеджер')))
          insert  into  Продажи  values  (@IDАвтомобиля,  @ФИОСотрудника,
@ФИОКлиента, @ДатаПродажи, @СтоимостьПродажи)
          else print 'Продажа невозможна'
     go

     insert into Продажи values (19, 'Володин В. В.', 'Бергер Б. Б.',
'11.09.11', 500000)
     insert into Продажи values (19, 'Чугунов Ч. Ч.', 'Бергер Б. Б.',
'11.13.11', 500000)
     insert into Продажи values (19, 'Абрамов А. А.', 'Бергер Б. Б.',
'11.13.11', 500000)
     insert into Продажи values (19, 'Сидоров С. С.', 'Бергер Б. Б.',
'11.13.11', 500000)
     insert into Продажи values (19, 'Володин В. В.', 'Бергер Б. Б.',
'11.13.11', 500000)
```

4. Создание триггера, контролирующего то, чтобы дата сервисного обслуживания была позже даты продажи и чтобы сервисное обслуживание осуществляли только сотрудники, имеющие должность Автомеханик:

```
create trigger Контроль_Обслуживания on Обслуживание
     instead of insert, update
     as
     declare @IDАвтомобиля int, @ФИОСотрудника nvarchar(50),
@ДатаОбслуживания datetime, @ПричинаОбращения nvarchar(50),
                @ДатаПродажи datetime
     select                                @IDАвтомобиля=IDАвтомобиля,
@ФИОСотрудника=ФИОСотрудника, @ДатаОбслуживания=ДатаОбслуживания,
                @ПричинаОбращения=ПричинаОбращения from inserted
     select    @ДатаПродажи=ДатаПродажи    from    Продажи    where
Продажи.IDАвтомобиля=@IDАвтомобиля
     if (@IDАвтомобиля in (select IDАвтомобиля from Продажи)) and
(@ДатаОбслуживания >= @ДатаПродажи) and
                          (@ФИОСотрудника in (select ФИОСотрудника
from Сотрудники where [№Офиса]<>1 and Должность='Автомеханик'))
     insert    into    Обслуживание    values    (@IDАвтомобиля,
@ФИОСотрудника, @ДатаОбслуживания, @ПричинаОбращения)
     else print 'Обслуживание невозможно'
go

insert into Обслуживание values (19, 'Юшин Ю. Ю.', '11.25.11',
'Покраска')
insert into Обслуживание values (9, 'Юшин Ю. Ю.', '11.01.11',
'Покраска')
insert into Обслуживание values (9, 'Кузьмин К. К.', '11.25.11',
'Покраска')
insert into Обслуживание values (9, 'Юшин Ю. Ю.', '11.25.11',
'Покраска')
```