

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФГАОУ ВО «СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
ИНСТИТУТ МАТЕМАТИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ
ИМЕНИ ПРОФЕССОРА Н. И. ЧЕРВЯКОВА
КАФЕДРА ПРИКЛАДНОЙ ИНФОРМАТИКИ

КУРСОВАЯ РАБОТА

по дисциплине
«БАЗЫ ДАННЫХ»

на тему:

Разработка базы данных «Интернет-магазин»

Выполнил:

Говоруха Никита Александрович

Студент(ка) 3 курса группы ПИН-б-о-181

Направления подготовки

09.03.03 Прикладная информатика

очной формы обучения

(подпись)

Руководитель работы:

(ФИО, должность, кафедра)

Работа допущена к защите _____
(подпись руководителя)

(дата)

Работа выполнена и
защищена с оценкой _____ Дата защиты _____

Члены комиссии: _____
(должность) (подпись) (И.О. Фамилия)

Ставрополь, 2020 г.

УТВЕРЖДАЮ
Зав. кафедрой ПИ
Азаров И.В.

Институт математики и информационных технологий имени профессора Н. И. Червякова
Кафедра прикладной информатики
Направление 09.03.03 Прикладная информатика
Профиль Прикладная информатика в экономике

ЗАДАНИЕ
на курсовую работу

студента: Говоруха Никиты Александровича
по дисциплине Базы данных

1. Тема работы: Разработка базы данных «Интернет-магазин»
2. Цель закрепление теоретических основ проектирования, создания и использования реляционных баз данных, получение практических навыков создания реляционных баз данных с помощью СУБД на примере SQL Management Studio 2016.
3. Рассмотреть виды СУБД, оценить перспективы развития СУБД, спроектировать базу данных на примере интернет магазина.
4. Перечень подлежащих разработке вопросов:
 - а) Физические модели данных, физическая запись (страница-вид физической записи). Последовательный файл, списковая структура, индексирование, В-дерево, хэш-функция
 - б) по практической части: проектирование базы данных «Интернет-магазин. Заказ товаров в интернете» с помощью среды Management Studio 2016.
5. Исходные данные:
 - а) по литературным источникам: Парфенов Ю.П. Постреляционные хранилища данных. Учебное пособие – УФУ, 2017. 122 с.
 - б) по вариантам, разработанным преподавателем _____
- в) иное _____
6. Список рекомендуемой литературы: Андон Ф., Резниченко В. Язык запросов SQL. Учебный курс СПб. Питер, Киев: BHV, 2006. — 416 с, Власова О.В. SQL Учебное пособие. – Ярославль: ЯрГУ, 2011. – 136 с., Дьяков И.А. Базы данных. Язык SQL Учебное пособие. — Тамбов: ТГТУ, 2004. — 80 с.
7. Контрольные сроки представления отдельных разделов курсовой работы:

25 % - _____	« ____ » _____	2020г.
50 % - _____	« ____ » _____	2020г.

75 % - _____ « ____ » _____ 2020г.
100 % - _____ « ____ » _____ 2020г.
8. Срок защиты студентом курсовой работы « ____ » _____ 2020г.
Дата выдачи задания “ ____ ” _____ 2020г.
Руководитель курсовой работы

(ученая степень, звание)(личная подпись)(инициалы, фамилия)

Задание принял к исполнению студент _____ формы обучения ____
курса ____ группы _____
(личная подпись) *(инициалы, фамилия)*

Содержание

Введение	5
1 Теоретическая часть	7
1.1 Физическая организация БД	7
1.1.1 Последовательная организация файлов	7
1.1.2 Индексно-последовательная организация файлов	7
1.1.3 Организация файлов с произвольным доступом	8
1.2 Индексные файлы в реляционных БД.....	9
1.3 Применение адресных указателей в БД.....	11
1.4 Целостность БД	13
2 Практическая часть	15
2.1 Описание предметной области. Постановка задачи.....	15
2.2 Выбор СУБД.....	15
2.3 Построение отношения в 1НФ.....	16
2.4 Формирование минимального множества функциональных зависимостей в отношении	17
2.6 Избавление от частичных зависимостей и перевод отношений в 2НФ	18
2.7 Определение существования транзитивных зависимостей в отношении и перевод всех отношений в 3НФ.....	19
2.8 ER-диаграмма предметной области	21
2.9 Создание базы данных в MS SQL Server	22
2.10 Создание диаграммы БД.....	22
2.11 Наполнение БД тестовыми данными. Примеры запросов на добавление, модификацию и удаление данных.....	23
2.12 Создание ограничения	27
2.13 Создание триггера	28
2.14 Создание представления со списком товаров, заказанных определенным клиентом.....	29
2.15 Создание хранимых процедур (ХП).....	30
Заключение	35
Список литературы	36

Введение

В настоящее время практически во всех организациях используются компьютеры для хранения и обработки служебной информации. Эта информация содержится в базах данных. Базы данных играют особую роль в современном мире. Все, с чем мы ежедневно сталкиваемся в жизни, скорее всего, зарегистрировано в той или иной базе. Умение работать с базами данных является одним из важнейших навыков в работе с компьютером, а специалисты этой области всегда окажутся востребованными. На сегодняшний день является актуальной обработка данных с помощью сетевых баз данных, поскольку с быстрым развитием глобальной сети.

Интернет возрастает необходимость в обмене данными на расстоянии. Иерархические базы данных продолжают использовать в тех случаях, если они заполнены большим количеством уникальных данных. Однако, в отдельных специальных задачах, где присутствуют только связи «один-ко-многим» иерархическая модель может оказаться полезной благодаря эффективному использованию памяти ЭВМ и быстрому выполнению операций над данными. В дисциплине баз данных рассмотрение иерархической и сетевой моделей является актуальным, так как они являются основополагающими.

Цели курсовой работы:

- формирование навыков научно-исследовательской работы;
- повышение уровня профессиональной подготовки;
- углубление знаний по учебной дисциплине; развитие интереса и навыков самостоятельной работы с научной и справочной литературой;
- разработка БД «Интернет магазин».

Для достижения поставленных целей были поставлены следующие задачи:

1. Изучить данные модели баз данных;
2. Сравнить структуру моделей данных;

3. Оценить продуктивность их работы в разных сферах;
4. Выявить достоинства и недостатки;
5. Спроектировать и разработать БД.

Объектом курсовой работы являются информационные процессы предметной области, а предметом курсовой работы – информационная платформа Microsoft SQL Server 2012. Практическая значимость курсовой работы заключается в проектировании, нормализации и реализации базы данных. В ходе работы применяются знания о концептуальном моделировании, нормализации, физическом моделировании и программировании на языке SQL, а также практические умения работы СУБД и программами графического моделирования.

1. Теоретическая часть

1.1 Физическая организация БД

Программист воспринимает файл в виде набора однородных записей. Запись — это наименьший элемент данных, который может быть обработан как единое целое прикладной программой при обмене с внешним устройством. Причем в большинстве ОС размер записи равен одному байту. В то время как приложения оперируют записями, физический обмен с устройством осуществляется большими единицами (обычно блоками). Поэтому записи объединяются в блоки для вывода и разблокируют - для ввода [2].

1.1.1 Последовательная организация файлов

Простейший вариант - так называемый последовательный файл. То есть файл является последовательностью записей. Поскольку записи, как правило, однобайтовые, файл представляет собой неструктурированную последовательность байтов.

Обработка подобных файлов предполагает последовательное чтение записей от начала файла, причем конкретная запись определяется ее положением в файле. Такой способ доступа называется последовательным (модель ленты). Если в качестве носителя файла используется магнитная лента, то так и делается. Текущая позиция считывания может быть возвращена к началу файла (rewind) [2].

1.1.2 Индексно-последовательная организация файлов

Отсортированный файл данных с первичным индексом называется индексированным последовательным файлом, или индексно-последовательным файлом [2]. Эта структура является компромиссом между файлами с полностью последовательной и полностью произвольной

организацией. В таком файле записи могут обрабатываться как последовательно, так и выборочно, с произвольным доступом, осуществляемым на основе поиска по заданному значению ключа с использованием индекса. Индексированный последовательный файл имеет более универсальную структуру, которая обычно включает следующие компоненты:

- 1) первичная область хранения;
- 2) отдельный индекс или несколько индексов;
- 3) область переполнения.

Обычно большая часть первичного индекса может храниться в оперативной памяти, что позволяет обрабатывать его намного быстрее. Для ускорения поиска могут применяться специальные методы доступа, например метод бинарного поиска. Основным недостатком использования первичного индекса (как и при работе с любым другим отсортированным файлом) является необходимость соблюдения последовательности сортировки при вставке и удалении записей. Эти проблемы усложняются тем, что требуется поддерживать порядок сортировки как в файле данных, так и в индексном файле. В подобном случае может использоваться метод, заключающийся в применении области переполнения и цепочки связанных указателей, аналогично методу, используемому для разрешения конфликтов в хешированных файлах.

1.1.3 Организация файлов с произвольным доступом

В реальной практике файлы хранятся на устройствах прямого (random) доступа, например, на дисках, поэтому содержимое файла может быть разбросано по разным блокам диска, которые можно считывать в произвольном порядке. Причем номер блока однозначно определяется позицией внутри файла.

Естественно, что в этом случае для доступа к середине файла просмотр всего файла с самого начала не обязателен. Для специфицирования места, с которого надо начинать чтение, используются два способа: с начала или с текущей позиции, которую дает операция seek. Файл, байты которого могут быть считаны в произвольном порядке, называется файлом прямого доступа.

Таким образом, файл, состоящий из однобайтовых записей на устройстве прямого доступа, - наиболее распространенный способ организации файла. Базовыми операциями для такого рода файлов являются считывание или запись символа в текущую позицию. В большинстве языков высокого уровня предусмотрены операторы посимвольной пересылки данных в файл или из него.

Подобную логическую структуру имеют файлы во многих файловых системах, например, в файловых системах ОС Unix. ОС не осуществляет никакой интерпретации содержимого файла. Эта схема обеспечивает максимальную гибкость и универсальность. С помощью базовых системных вызовов (или функций библиотеки ввода/вывода) пользователи могут как угодно структурировать файлы. В частности, многие СУБД хранят свои базы данных в обычных файлах.

1.2 Индексные файлы в реляционных БД

Для реализации в СУБД таких функций как автоматическая сортировка записей, контроль за отсутствием повторений значений в ключевых полях записей и повышение скорости выполнения операций поиска в таблице применяют индексирование.

Основным преимуществом использования индексирования является значительное ускорение процесса выборки или извлечения данных, основным недостатком – замедление процесса обновления данных, т. к. при каждом добавлении новой записи в индексированный файл потребуется также добавить новый индекс в индексный файл.

Индекс (index) – средство ускорения операции поиска записей в таблице, а также выполнения других операций, использующих поиск: извлечение, модификация, сортировка и т. д. Индексный файл (index file) – это файл, в котором хранится информация индекса [4]. Он является файлом особого типа, в котором каждая запись состоит из двух значений: данных и указателя номера записи. При этом данные необходимы для индексного поля из индексированного файла, а указатель – для связывания с соответствующей записью индексированного файла.

На практике для создания индекса для некоторой таблицы базы данных пользователь указывает поле таблицы, которое требует индексации.

Термин «индекс» тесно связан с понятием «ключ», хотя между ними есть и некоторое отличие.

Если индексирование организовано на основе ключевого поля, то индекс называется первичным. Ключевые поля, как правило, индексируются автоматически.

Если индекс организован на основе другого поля, то он называется вторичным. Индекс, организованный на основе ключевого поля или другого ключа, называется уникальным.

На практике индексы можно использовать двумя разными способами:

- последовательного доступа к индексированному файлу, т. е. в последовательности, заданной значениями индексного поля;
- прямого доступа к отдельным записям индексированного файла на основе заданного значения индексного поля [4].

Хранимый файл может иметь несколько индексов. Часто индекс создают на основе комбинации двух или более полей. При использовании пары индексов требуется два отдельных просмотра, скорость выполнения запроса может сильно зависеть от последовательности выполнения отдельных просмотров по индексам.

Основной целью использования индекса является ускорение процесса извлечения данных, за счет уменьшения числа дисковых операций ввода-вывода, для чего используются указатели.

Индекс, в котором не содержатся указатели на все записи индексированного файла, называется неплотным. Одним из преимуществ неплотных индексов является их малый размер по сравнению с плотными индексами. т. к., они содержат меньшее число записей. Это часто позволяет просматривать содержимое БД с большей скоростью, но нельзя выполнить проверку наличия некоторого значения.

1.3 Применение адресных указателей в БД

Курсор является одним из объектов адресных указателей.

Курсор — ссылка на контекстную область памяти. В некоторых реализациях языка программирования SQL (Oracle, Microsoft SQL Server) — получаемый при выполнении запроса результирующий набор и связанный с ним указатель текущей записи. Я бы сказал, что курсор — это виртуальная таблица, которая представляет собой альтернативное хранилище данных. При этом курсор, позволяет обращаться к своим данным, как к данным обычного массива. Используются курсоры в хранимых процедурах [2].

Команды манипулирования данными SELECT, UPDATE, DELETE работают сразу с группами строк. Эти группы, вплоть до отдельных строк, можно выбрать с помощью опции WHERE. А если надо перебрать строки некоторой таблицы последовательно, одну за другой? На этот случай в языке SQL существуют курсоры. Курсор (current set of record) – временный набор строк, которые можно перебирать последовательно, с первой до последней.

При работе с курсорами используются следующие команды.

Объявление курсора:

DECLARE имя_курсора CURSOR FOR SELECT текст_запроса

Любой курсор создается на основе некоторого оператора SELECT.

Открытие курсора:

OPEN имя_курсора

Для того чтобы с помощью курсора можно было читать строки, его надо обязательно открыть.

Чтение следующей строки из курсора:

FETCH имя_курсора INTO список_переменных

Переменные в списке должны быть в том же количестве и того же типа, что и столбцы курсора.

Глобальная переменная @@FETCH_STATUS принимает ненулевое значение, если строк в курсоре больше нет. Если же набор строк еще не исчерпан, то @@FETCH_STATUS равна нулю, и оператор FETCH переписывает значения полей из текущей строки в переменные.

Закрытие курсора:

CLOSE имя_курсора

Для удаления курсора из памяти используется команда

DEALLOCATE имя_курсора

Для иллюстрации использования курсора создадим процедуру, которая будет выбирать данные из одной таблицы, перебирать их в курсоре анализируя, есть ли такие данные во второй таблице и вставлять в третью таблицу, если данные записи удовлетворяют определенным критериям.

```
CREATE PROCEDURE [dbo].[MyProcedure] AS
```

```
DECLARE @ID INT
```

```
DECLARE @QUA INT
```

```
DECLARE @VAL VARCHAR (500)
```

```
DECLARE @NAM VARCHAR (500)
```

```
/*Объявляем курсор*/
```

```
DECLARE @CURSOR CURSOR
```

```
/*Заполняем курсор*/
```

```
SET @CURSOR = CURSOR SCROLL
```

```
FOR
```

```
SELECT INDEX, QUANTITY, VALUE, NAME
```

```
FROM My_First_Table WHERE QUANTITY > 1
```

```
/*Открываем курсор*/
```

```

OPEN @CURSOR
/*Выбираем первую строку*/
FETCH NEXT FROM @CURSOR INTO @ID, @QUA, @VAL, @NAM
/*Выполняем в цикле перебор строк*/
WHILE @@FETCH_STATUS = 0
BEGIN

    IF NOT EXISTS(SELECT VAL FROM My_Second_Table WHERE ID=@ID)
    BEGIN
        /*Вставляем параметры в третью таблицу если условие соблюдается*/
        INSERT INTO My_Third_Table (VALUE, NAME) VALUE(@VAL, @NAM)
    END

    /*Выбираем следующую строку*/
    FETCH NEXT FROM @CURSOR INTO @ID, @QUA, @VAL, @NAM
END
CLOSE @CURSOR

```

1.4 Целостность БД

Целостность базы данных (database integrity) — соответствие имеющейся в базе данных информации её внутренней логике, структуре и всем явно заданным правилам. Каждое правило, налагающее некоторое ограничение на возможное состояние базы данных, называется ограничением целостности (integrity constraint) [5].

Примеры правил: вес детали должен быть положительным; количество знаков в телефонном номере не должно превышать 15; возраст родителей не может быть меньше возраста их биологического ребёнка и т. д.

Очевидно, что ограничения должны быть формально объявлены для СУБД, после чего СУБД должна предписывать их выполнение. Объявление ограничений сводится просто к использованию соответствующих средств языка базы данных, а соблюдение ограничений осуществляется с помощью контроля со стороны СУБД над операциями обновления, которые могут нарушить эти ограничения, и запрещения тех операций, которые их действительно нарушают. При первоначальном объявлении ограничения

система должна проверить, удовлетворяет ли ему в настоящий момент база данных. Если это условие не соблюдается, ограничение должно быть отвергнуто; в противном случае оно принимается (то есть записывается в каталог системы) и начиная с этого момента соблюдается.

Механизмы обеспечения целостности являются одной из составляющих концепции модели данных.

Целостность БД не гарантирует достоверности (истинности) содержащейся в ней информации, но обеспечивает по крайней мере правдоподобность этой информации, отвергая заведомо невероятные, невозможные значения. Таким образом, не следует путать целостность (непротиворечивость) БД с истинностью БД. Истинность и непротиворечивость — не одно и то же.

Достоверность (или истинность) есть соответствие фактов, хранящихся в базе данных, реальному миру. Очевидно, что для определения достоверности БД требуется обладание полными знаниями как о содержимом БД, так и о реальном мире. Для определения целостности БД требуется лишь обладание знаниями о содержимом БД и о заданных для неё правилах. Поэтому СУБД не может гарантировать наличие в базе данных только истинных высказываний; все, что она может сделать, — это гарантировать отсутствие каких-либо данных, вызывающих нарушение ограничений целостности (то есть гарантировать то, что она не содержит каких-либо данных, не совместимых с этими ограничениями).

Из того, что данные являются правильными, следует, что они непротиворечивы (но не обратное), а из того, что данные противоречивы, следует, что они неправильны (но не обратное). Здесь под словом «правильные» подразумевается, что в базе данных содержатся правильные данные тогда и только тогда, когда она полностью отражает истинное состояние дел в реальном мире.

2 Практическая часть

2.1 Описание предметной области. Постановка задачи

Интернет-магазин предоставляет сервисы заказа товаров. Требуется обеспечить хранение и обработку следующих данных:

1. Информация о персональных данных клиентов с указанием, как минимум, ФИО, даты рождения, пола и т.п.
2. Информация о товарах: категория, наименование, масса, габаритные размеры, состав компонент, цена.
3. Информация о заказе товаров: дата операции, клиент, заказанные товары, количество, стоимость.

2.2 Выбор СУБД

При выполнении данной курсовой работы для построения базы данных мы выбрали СУБД MS Server, так как Microsoft SQL Server является довольно мощной системой. Помимо стандартных для СУБД функций, SQL Server 2012 содержит большой набор интегрированных служб по анализу данных. В данной работе мы использовали версию MS Server, так как именно эта версия данного программного продукта является стабильной, проверенной временем, достаточно функциональной.

2.3 Построение отношения в 1НФ

Начнем с реализации физической модели данных. Спроектируем таблицу и опишем все атрибуты. При составлении таблицы придерживаемся требований 1НФ, а именно:

А) нет повторяющихся строк;

Б) нет упорядочивания столбцов слева направо (порядок столбцов не несет никакой информации);

В) нет упорядочивания строк сверху вниз (порядок строк не несет в себе никакой информации);

Г) каждое пересечение строки и столбца содержит ровно одно значение из соответствующего домена;

Д) все атрибуты являются атомарными (неделимыми);

Е) все столбцы являются обычными;

Отношение «Orders» в первой нормальной форме (рисунок 1).

IdOrder
OrDate
OrAddress
IdClient
CName
CLName
CPhone
CBirthOfDate
CSEX
IdProduct
PName
PDescription
PPrice
PHeight
PWidth
PDepth
IdCategory
CatName
IdManufacture
MaName
MaDescription
IdMaterial
MatName
count
TotalPrice

Рисунок 1 - Отношение "ORDER" в первой нормальной форме

2.4 Формирование минимального множества функциональных зависимостей в отношении

Определим следующие функциональные зависимости:

$\text{IdOrder, IdProduct} \rightarrow \dots$

$\text{IdClient, OrDate, IdProduct} \rightarrow \dots$

$\text{IdOrder} \rightarrow \text{OrDate}$

$\text{IdOrder} \rightarrow \text{OrAddress}$

$\text{IdOrder} \rightarrow \text{IdClient}$

$\text{IdClient} \rightarrow \text{CLName}$

$\text{IdClient} \rightarrow \text{CName}$

$\text{IdClient} \rightarrow \text{CPhone}$

$\text{IdClient} \rightarrow \text{CBirthOfDate}$

$\text{IdClient} \rightarrow \text{CSEX}$

$\text{IdProduct} \rightarrow \text{PName}$

$\text{IdProduct} \rightarrow \text{PDescription}$

$\text{IdProduct} \rightarrow \text{PPrice}$

$\text{IdProduct} \rightarrow \text{PHeight}$

$\text{IdProduct} \rightarrow \text{PWidht}$

$\text{IdProduct} \rightarrow \text{PDepth}$

$\text{IdProduct} \rightarrow \text{IdCategory}$

$\text{IdProduct} \rightarrow \text{IdManufacture}$

$\text{IdProduct} \rightarrow \text{MaName}$

$\text{IdProduct} \rightarrow \text{MaDescription}$

$\text{IdProduct} \rightarrow \text{IdMaterial}$

$\text{IdProduct} \rightarrow \text{MatName}$

$\text{IdCategory} \rightarrow \text{CatName}$

$\text{IdManufacture} \rightarrow \text{MaName}$

$\text{IdManufacture} \rightarrow \text{MaDescription}$

$\text{IdMaterial} \rightarrow \text{MatName}$

$\text{PPrice} \rightarrow \text{TotalPrice}$

2.5 Определение потенциальных ключей, первичных ключей отношений

Потенциальный ключ – представляет собой минимальный суперключ отношения, т. е. набор атрибутов, который удовлетворяет ряду условий [2]:

- неприводимость (не может быть сокращен, содержит минимально возможный набор атрибутов);
- уникальность (должен иметь уникальные значения вне зависимости от изменения строки);
- наличие значения (не должен иметь значения NULL, обязательно должен иметь значение).

Исходя из этого выделим следующие потенциальные ключи:

IdOrder, IdProduct, IdClient

2.6 Избавление от частичных зависимостей и перевод отношений в 2НФ

Отношение в 2НФ это отношение, которое находится в первой нормальной форме и каждый атрибут которого, не входящий в состав первичного ключа, характеризуется полной функциональной зависимостью от этого первичного ключа.

Исходя из минимальных множеств функциональных зависимостей можно выделить следующие отношения:

Отношение «CLIENT» (рисунок 2).

IdClient	int	<input type="checkbox"/>
CLName	nvarchar(25)	<input type="checkbox"/>
CName	nchar(15)	<input type="checkbox"/>
CPhone	int	<input type="checkbox"/>
CDateOfBirth	date	<input checked="" type="checkbox"/>
CSEX	nvarchar(3)	<input checked="" type="checkbox"/>

Рисунок 2 – таблица Client

Отношение «PRODUCT» (рисунок 3).

IdProduct	int	<input type="checkbox"/>
PName	nchar(10)	<input type="checkbox"/>
PDescription	text	<input type="checkbox"/>
PPrice	float	<input type="checkbox"/>
PWidth	float	<input type="checkbox"/>
PHeight	float	<input type="checkbox"/>
PDepth	float	<input type="checkbox"/>
IdCategory	int	<input type="checkbox"/>
CatName	nvarchar(25)	<input type="checkbox"/>
IdManufacture	int	<input type="checkbox"/>
MaName	nvarchar(25)	<input type="checkbox"/>
MaDescription	text	<input checked="" type="checkbox"/>
IdMaterial	int	<input type="checkbox"/>
MatName	nvarchar(25)	<input type="checkbox"/>

Рисунок 3 – таблица Product

Отношение «ORDER» (рисунок 4).

IdOrder	int	<input type="checkbox"/>
OrDate	date	<input checked="" type="checkbox"/>
OrAddress	nvarchar(100)	<input type="checkbox"/>
IdCleint	int	<input type="checkbox"/>
BoxWidht	float	<input checked="" type="checkbox"/>
BoxHeight	float	<input checked="" type="checkbox"/>
BoxDepth	float	<input checked="" type="checkbox"/>

Рисунок 4 – таблица ORDER

Отношение «ORDERDETAILS» (рисунок 5).

IdOrder	int	<input type="checkbox"/>
IdProduct	int	<input type="checkbox"/>
Count	int	<input checked="" type="checkbox"/>
TotalPrice	float	<input checked="" type="checkbox"/>

Рисунок 5 – таблица ORDERDETAILS

2.7 Определение существования транзитивных зависимостей в отношении и перевод всех отношений в 3НФ

При подробном рассмотрении были выведены следующие транзитивные функциональные зависимости:

1) $\text{IdProduct} > \text{IdCategory}, \text{CatName}$

$\text{IdCategory} > \text{CatName}$

Исходя из этого сформируем отношение по теореме Хеза (рисунок 6).

IdCategory	int	<input type="checkbox"/>
CatName	nvarchar(25)	<input checked="" type="checkbox"/>

Рисунок 6 – таблица CATEGORY

2) $\text{IdProduct} > \text{IdManufacture}, \text{MaName}, \text{MaDescription}$

$\text{IdManufacture} > \text{MaName}, \text{MaDescription}$

Исходя из этого сформируем отношение по теореме Хеза (рисунок 7).

IdManufacture	int	<input type="checkbox"/>
MaName	nvarchar(25)	<input type="checkbox"/>
MaDescription	text	<input checked="" type="checkbox"/>

Рисунок 7 – таблица MANUFACTURE

3) $\text{IdGood} > \text{IdMaterial}, \text{MatName}$

$\text{IdMaterial} > \text{MatName}$

Исходя из этого сформируем отношение по теореме Хеза (рисунок 8).

IdMaterial	int	<input type="checkbox"/>
MatName	nvarchar(25)	<input type="checkbox"/>

Рисунок 8 – таблица MATERIAL

После всех преобразований отношение PRODUCT будет иметь следующий вид (рисунок 9).

IdProduct	int	<input type="checkbox"/>
PName	nvarchar(30)	<input type="checkbox"/>
PDescription	text	<input type="checkbox"/>
PPrice	float	<input type="checkbox"/>
PWidth	float	<input type="checkbox"/>
PHeight	float	<input type="checkbox"/>
PDepth	float	<input type="checkbox"/>
IdCategory	int	<input type="checkbox"/>
IdManufacture	int	<input type="checkbox"/>
IdMaterial	int	<input type="checkbox"/>

Рисунок 9 – таблица GOOD после преобразований

Все наши отношения находятся в Нормальной форме Бойса-Кодда, т. к. все нетривиальные и неприводимые слева функциональные зависимости обладают потенциальным ключом в качестве детерминанта. А если все отношения находятся в НФБК, следовательно они находятся и в 3НФ.

2.8 ER-диаграмма предметной области

ENTITY RELATIONAL (ER) MODEL — это концептуальная модель модели данных высокого уровня. ER моделирование помогает систематически анализировать требования к данным для создания хорошо спроектированной базы данных. Модель сущности-отношения представляет сущности реального мира и отношения между ними [2] (рисунок 10).

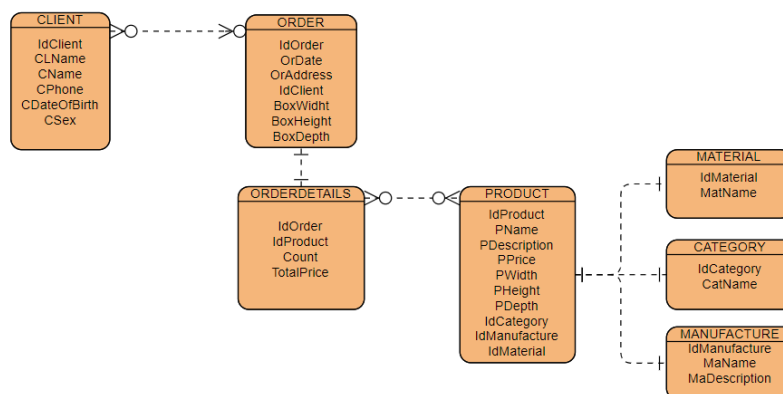


Рисунок 10 – ER-диаграмма

2.9 Создание базы данных в MS SQL Server

Создадим и именуем таблицы БД; именуем атрибуты таблиц и выберем типы данных; определим первичные и внешние ключи таблиц и выберем типы данных. Где нужно определяем ограничения атрибутов и значения по умолчанию. Т. к. мы создали БД и таблицы на этапе нормализации то пропустим этот пункт и выведем все созданные таблицы в следующем пункте.

2.10 Создание диаграммы БД

Определим правила поддержания ссылочной целостности системы ключей (рисунок 11).

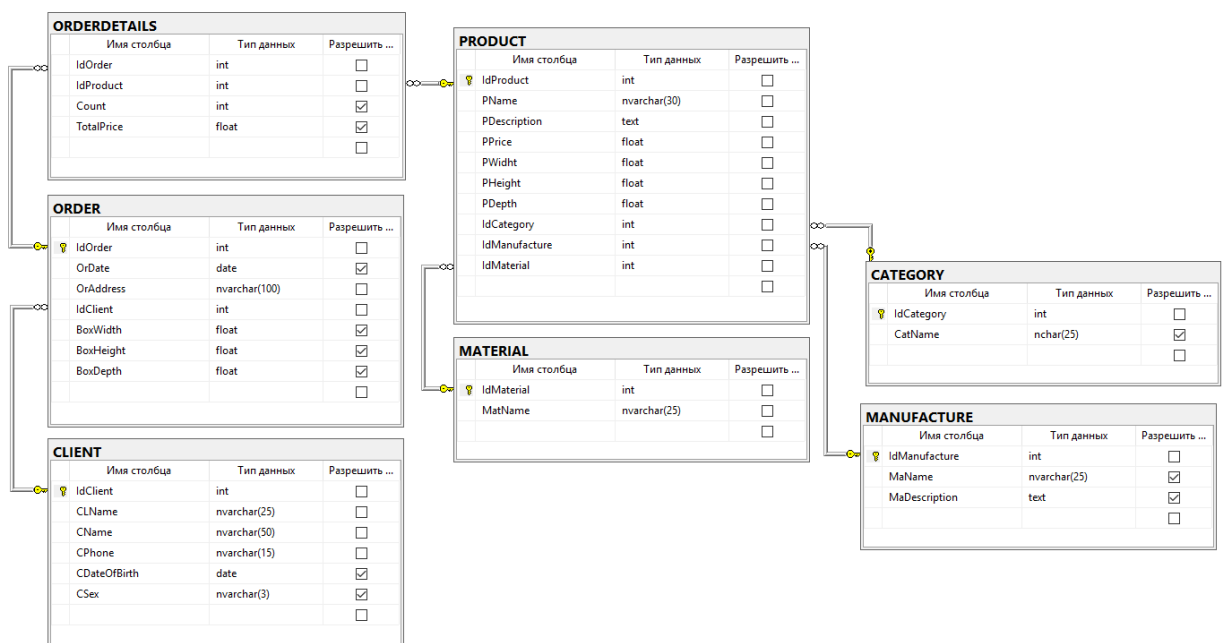


Рисунок 11 – диаграмма базы данных

2.11 Наполнение БД тестовыми данными. Примеры запросов на добавление, модификацию и удаление данных

Создадим хранимую процедуру для добавления пользователя в таблицу CLIENT (рисунок 12):

```
USE [Online store]

GO
CREATE PROCEDURE AddClient
    @LName NVARCHAR(25),
    @Name NVARCHAR(15),
    @PhoneNumber INT,
    @DateOfBirth DATE,
    @Sex NVARCHAR(3)
AS
INSERT INTO CLIENT(CLName, CName, CPhone, CDateOfBirth, CSex)
VALUES (@LName, @Name, @PhoneNumber, @DateOfBirth, @Sex)
```

Рисунок 12 – хранимая процедура для добавления пользователя

С помощью этой хранимой процедуры напомним таблицу и получим следующий результат (рисунок 13).

	IdClient	CLName	CName	CPhone	CDateOfBirth	CSex
1	1	Иванов	Сергей	89193337373	1998-01-01	муж
2	2	Арзамасов	Михаил	89883766512	2001-03-12	муж
3	3	Блинова	Арина	89069873737	1989-04-27	жен
4	4	Катышкин	Антон	89058344312	1984-11-12	муж
5	5	Водопоев	Дмитрий	89038263795	2000-06-12	муж
6	6	Юрьева	Светлана	89046278270	1996-05-16	жен
7	7	Михайлова	Екатерина	89191234567	1997-03-14	жен
8	8	Баранов	Олег	89001237654	2001-06-22	муж
9	9	Малахов	Даниил	89188234587	2000-08-04	муж
10	10	Балабанова	Ольга	89997534838	1994-02-27	жен

Рисунок 13 – отношение CLIENT, созданное с помощью хранимой процедуры

С помощью запросов заполним таблицы «MATERIAL», «MANUFACTURE» и «CATEGORY». Получим следующий результат (рисунок 14).

	IdCategory	CatName		IdManufacture	MaName	MaDescription		IdMaterial	MatName
1	1	Смартфон					1	1	Пластик
2	2	Планшет	1	1	Apple	NULL	2	2	Метал
3	3	Плеер	2	2	Samsung	NULL	3	3	Стекло
4	4	Наушники	3	3	Xiaomi	NULL			
5	5	Чехол	4	4	Huawei	NULL			

Рисунок 14 – отношения MATERIAL, MANUFACTURE и CATEGORY

Создадим хранимую процедуру для добавления товара в таблицу PRODUCT (рисунок 15).

```

USE [Online store]
GO
CREATE PROCEDURE AddProduct
    @PName NVARCHAR(30),
    @PDescription TEXT,
    @PPrice FLOAT,
    @PWidht FLOAT,
    @PHeight FLOAT,
    @PDepth FLOAT,
    @IdCategory INT,
    @IdManufacture INT,
    @IdMaterial INT
AS
INSERT INTO PRODUCT(PName, PDescription, PPrice, PWidht, PHeight, PDepth, IdCategory, IdManufacture, IdMaterial)
VALUES (@PName, @PDescription, @PPrice, @PWidht, @PHeight, @PDepth, @IdCategory, @IdManufacture, @IdMaterial)

```

Рисунок 15 – создание хранимой процедуры для добавления товара

С помощью этой хранимой процедуры наполним таблицу и получим следующий результат (рисунок 16).

IdProduct	PName	PDescription	PPrice	PWidht	PHeight	PDepth	IdCategory	IdManufacture	IdMaterial
2	Iphone 7	Хороший сма...	25000	158	78	7	1	1	2
5	Iphone 6	Бюджетный с...	20000	138	67	7	1	1	2
7	Iphone 8	Смартфон с б...	30000	134	67	7	1	1	1
8	Iphone X	Смартфон нов...	40000	143	71	8	1	1	3
9	Iphone 11	Новейший см...	50000	150	76	8	1	1	3
10	AirPods	Беспроводные...	10000	65	55	10	4	1	1
11	AirPods 2	Беспроводные...	12000	65	55	10	4	1	1
12	EarPods	Проводные на...	3000	80	65	15	4	1	1
13	Galaxy S20	Современный ...	60000	152	70	8	1	2	3
14	Mi 10	Технологичны...	35000	162	75	9	1	3	1

Рисунок 15 – отношение PRODUCT

С помощью запроса создадим первые заказы в нашей БД. Создание заказа с помощью запроса (рисунок 17).


```

USE [Online store]

GO
INSERT INTO [dbo].[ORDER]
    ([OrDate],
    [OrAddress],
    [IdClient],
    [BoxWidth],
    [BoxHeight],
    [BoxDepth])
VALUES
    ('2020-12-16',
    'г. Ставрополь ул. Космонавтов д. 3 кв. 12',
    3,
    100,
    12,
    25)

```

Рисунок 16 – создание заказа с помощью запроса

Создание деталей заказа с помощью запроса (рисунок 18).

```

USE [Online store]
GO
INSERT INTO [dbo].[ORDERDETAILS]
    ([IdOrder],
    [IdProduct],
    [Count],
    [TotalPrice])
VALUES
    (1,
    2,
    2,
    50000)

```

Рисунок 17 – создание деталей заказа в таблице ORDERDETAILS

Пример заполнения таблиц «ORDER» и «ORDERDETAILS» (рисунок 19).

	IdOrder	OrDate	OrAddress	IdClient	BoxWidth	BoxHeight	BoxDepth
1	1	2020-12-16	г. Ставрополь ул. Космонавтов д. 3 кв. 12	3	100	12	25
2	2	2020-12-21	г. Ставрополь ул. Павлова д. 25	5	180	13	22
3	3	2020-12-21	г. Ставрополь ул. Павлова д. 11	7	200	10	50

	IdOrder	IdProduct	Count	TotalPrice
1	1	2	2	50000
2	2	5	1	20000
3	3	13	1	60000

Рисунок 18 – Отношение ORDER и ORDERDETAILS

Теперь создадим хранимые процедуры для удаления клиента и обновления информации о товаре.

Процедура для удаления клиента из таблицы CLIENT (рисунок 20):

```
USE [Online store]
GO
ALTER PROCEDURE DeleteClient @IdClient INT
AS
DELETE FROM CLIENT
WHERE IdClient = @IdClient
EXEC DeleteClient 4
```

Рисунок 19 – процедура для удаления клиента

Результат выполнения хранимой процедуры (рисунок 21).

	IdClient	CLName	CName	CPhone	CDateOfBirth	CSEX
1	1	Иванов	Сергей	89193337373	1998-01-01	муж
2	2	Арзамасов	Михаил	89883766512	2001-03-12	муж
3	3	Блинова	Арина	89069873737	1989-04-27	жен
4	5	Водопоев	Дмитрий	89038263795	2000-06-12	муж
5	6	Юрьева	Светлана	89046278270	1996-05-16	жен
6	7	Михайлова	Екатерина	89191234567	1997-03-14	жен
7	8	Баранов	Олег	89001237654	2001-06-22	муж
8	9	Малахов	Даниил	89188234587	2000-08-04	муж
9	10	Балабанова	Ольга	89997534838	1994-02-27	жен

Рисунок 21 – отношение CLIENT после использования хранимой процедуры для удаления клиента по Id

Процедура для изменения товара из таблицы PRODUCT (рисунок 22):

```

USE [Online store]
GO
ALTER PROCEDURE UpdateProduct
@IdProduct INT,
@Price FLOAT
AS
UPDATE PRODUCT
SET PPrice = @Price
WHERE IdProduct = @IdProduct
EXEC UpdateProduct 14, 34999

```

Рисунок 22 – изменение цены товара с Id 14

Результат выполнения хранимой процедуры (рисунок 23).

	IdProduct	PName	PDescription	PPrice	PWidht	PHeight
1	2	Iphone 7	Хороший смартфон, вышел в 2016-ом году	25000	158	78
2	5	Iphone 6	Бюджетный смартфон	20000	138	67
3	7	Iphone 8	Смартфон с большим экраном	30000	134	67
4	8	Iphone X	Смартфон нового поколения	40000	143	71
5	9	Iphone 11	Новейший смартфон	50000	150	76
6	10	AirPods	Беспроводные наушники	10000	65	55
7	11	AirPods 2	Беспроводные наушники нового поколения	12000	65	55
8	12	EarPods	Проводные наушники	3000	80	65
9	13	Galaxy S20	Современный смартфон	60000	152	70
10	14	Mi 10	Технологичный смартфон	34999	162	75

Рисунок 23 – Таблица PRODUCT после выполнения хранимой процедуры, которая меняет цену товара по ID

2.12 Создание ограничения

Создадим ограничение на единовременный заказ товаров меньше минимально разрешенной общей стоимости. Реализуем это ограничение через скалярную функцию. Сначала создадим функцию, которая будет подсчитывать общую сумму заказа (рисунок 24).

```

USE [Online store]
GO
ALTER FUNCTION [dbo].[TotalPriceCounter]
(@IdOrder INT)
RETURNS bit
BEGIN
DECLARE @TotalPrice FLOAT
SELECT @TotalPrice = SUM(TotalPrice)
FROM ORDERDETAILS
WHERE @IdOrder = IdOrder
IF (@TotalPrice < 5000)
RETURN 1
RETURN 0
END

```

Рисунок 24 – функция, которая ограничивает сумму заказа

Данная скалярная функция высчитывает сумму заказа по ID заказа, если сумма заказа меньше 5000, то функция возвращает 1, если больше – 0.

2.13 Создание триггера

Создадим триггер, актуализирующий информацию стоимости заказанных товаров. Допустим в таблице PRODUCT товары указаны без учета НДС 20%, тогда создадим триггер, который будет увеличивать TotalPrice на 20% за каждую единицу Count заказанного товара (рисунок 25).

```

USE [Online store]
GO
CREATE TRIGGER TotalPriceWithTax
ON [dbo].[ORDERDETAILS]
AFTER INSERT, UPDATE
AS
BEGIN
UPDATE ORDERDETAILS
SET ORDERDETAILS.TotalPrice = (ORDERDETAILS.TotalPrice * ORDERDETAILS.Count * 1.2)
WHERE ORDERDETAILS.IdOrder = (SELECT IdOrder FROM inserted)
END
Используем созданный триггер:
USE [Online store]
INSERT INTO ORDERDETAILS (IdOrder, IdProduct, Count, TotalPrice)
VALUES (4, 7, 1, 40000)
SELECT *
FROM ORDERDETAILS

```

Рисунок 25 – триггер, который увеличивает общую сумму на 20% за каждую единицу товара

Отношение, полученное после выполнения триггера (рисунок 26):

	IdOrder	IdProduct	Count	TotalPrice
1	1	2	2	50000
2	2	5	1	20000
3	4	7	1	48000

Рисунок 26 – Результат выполнения триггера, который увеличивает цену товара на 20%

2.14 Создание представление со списком товаров, заказанных определенным клиентом

Создадим представление со списком товаров, заказанных определенным клиентом (рисунок 27).

```
USE [Online store]

SELECT CLIENT.CLName, CLIENT.CName, PRODUCT.PName, PRODUCT.PPrice, ORDERDETAILS.Count, ORDERDETAILS.TotalPrice
FROM CLIENT INNER JOIN
[ORDER] ON dbo.CLIENT.IdClient = [ORDER].IdClient INNER JOIN
ORDERDETAILS ON dbo.[ORDER].IdOrder = ORDERDETAILS.IdOrder INNER JOIN
PRODUCT ON ORDERDETAILS.IdProduct = PRODUCT.IdProduct
```

Рисунок 27 – представление со списком товаров, заказанных клиентом

Результат представления (рисунок 28).

	CLName	CName	PName	PPrice	Count	TotalPrice
1	Блинова	Арина	Iphone 7	25000	2	50000
2	Водопоев	Дмитрий	Iphone 6	20000	1	20000
3	Юрьева	Светлана	Iphone 8	30000	1	48000

Рисунок 28 – представление, которое выводит список товаров, который заказал определенный клиент

2.15 Создание хранимых процедур (ХП)

а) Создадим ХП, которая по номеру клиента возвращает отсортированный список заказанных товаров за указанный период. Результат выполнения хранимой процедуры представлен на рисунке (рисунок 29).

```
CREATE PROCEDURE OrderBetweenDate
@beginDate date,
@endDate date
AS
BEGIN
SELECT c.CLName as 'Фамилия', c.CName as 'Имя', p.PName as 'Название товара',
o.OrDate as 'Дата заказа товара'
from [ORDER] o
JOIN [ORDERDETAILS] od ON o.IdOrder = od.IdOrder
JOIN [PRODUCT] p ON od.IdProduct = p.IdProduct
JOIN [CLIENT] c ON c.IdClient = o.IdClient
WHERE o.OrDate BETWEEN @beginDate AND @endDate
ORDER BY o.OrDate
END
```

Рисунок 209 – процедура, которая выводит список заказных товаров за указанный период

Вызовем полученную функцию (рисунок 30).

```
USE [Online store]
EXEC [OrderBeetwenDate] 3, '01-01-2020', '01-01-2021'
```

Рисунок 30– вызов функции OrderBetweenDate

Результат вызова (рисунок 31).

	Название товара	Кол-во купленного товара	Цена тоавара за одну штуку	Дата заказа товара
1	lphone 7	2	25000	2020-12-16

Рисунок 31 – отношение, которое показывает заказанные товары, за определенный промежуток времени

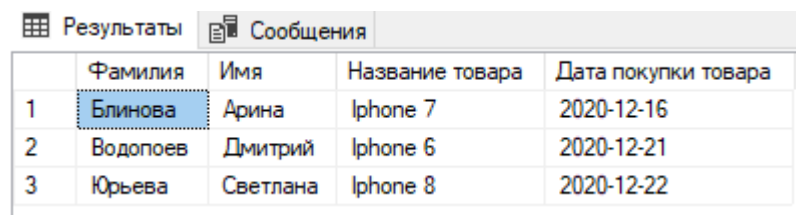
а) Создадим ХП, которая выводит список товаров, которые были заказаны в указанный период. Решим задачу с использованием 1) операции соединения и 2) подзапроса.

- 1) Реализуем хранимую процедуру с использованием операции соединения (рисунок 32).

```
CREATE PROCEDURE ProductsPerDate
@BeginDate DATE,
@EndDate DATE
AS
BEGIN
SELECT C.CLName AS 'Фамилия', C.CName AS 'Имя', P.PName AS 'Название товара',
O.OrDate AS 'Дата покупки товара'
FROM [ORDER] O
JOIN [ORDERDETAILS] Od ON O.IdOrder = Od.IdOrder
JOIN [PRODUCT] P ON Od.IdProduct = P.IdProduct
JOIN [CLIENT] C ON C.IdClient = O.IdClient
WHERE O.OrDate BETWEEN @BeginDate AND @EndDate
ORDER BY O.OrDate
END;
```

Рисунок 32 – реализация хранимой процедуры с использованием операции соединения

Вызовем полученную функцию (рисунок 33).



	Фамилия	Имя	Название товара	Дата покупки товара
1	Блинова	Арина	Iphone 7	2020-12-16
2	Водопоев	Дмитрий	Iphone 6	2020-12-21
3	Юрьева	Светлана	Iphone 8	2020-12-22

Рисунок 33 – результат хранимой процедуры, которая выводит список покупок за определенный промежуток времени

- 2) Реализуем хранимую процедуру с использованием подзапроса (рисунок 34).

```
CREATE PROCEDURE ProductsPerDateSubquery
@BeginDate DATE,
@EndDate DATE
AS
BEGIN
SELECT C.CLName AS 'Фамилия', C.CName AS 'Имя', P.PName AS 'Название товара',
O.OrDate AS 'Дата заказа товара'
FROM [CLIENT] C, [PRODUCT] P, [ORDER] O
WHERE P.IdProduct IN (SELECT OD.IdProduct FROM [ORDERDETAILS] OD
WHERE O.OrDate BETWEEN @BeginDate AND @EndDate
AND C.IdClient = O.IdClient
AND Od.IdOrder = O.IdOrder)
ORDER BY O.OrDate
END;
```

Рисунок 34 – реализация хранимой процедуры с использованием подзапроса

Вызовем полученную хранимую процедуру и получим результат вызова (рисунок 35).

EXEC ProductsPerDateSubquery '01-01-2020', '01-01-2021'				
Результаты		Сообщения		
	Фамилия	Имя	Название товара	Дата заказа товара
1	Блинова	Арина	Iphone 7	2020-12-16
2	Водопоев	Дмитрий	Iphone 6	2020-12-21
3	Юрьева	Светлана	Iphone 8	2020-12-22

Рисунок 35 – результат хранимой процедуры с использованием подзапроса, которая выводит список покупок за определенный промежуток времени

с) Создадим хранимую процедуру, для вывода отсортированного списка товаров с указанием суммарной выручки от их продажи (рисунок 36).

```
CREATE PROCEDURE AllSumForProduct
AS
BEGIN
SELECT P.PName, SUM(Od.TotalPrice) AS 'Суммарная выручка'
FROM [ORDERDETAILS] Od
JOIN [PRODUCT] P ON Od.IdProduct = P.IdProduct
GROUP BY P.PName
END;
```

Рисунок 21 – хранимая процедура AllSumForProduct, которая выводит отсортированный список товаров с указанием суммы выручки

Полученный результат от вызова процедуры AllSumForProduct, которая выводит отсортированный список товаров с указанием выручки (рисунок 37).

Результаты		Сообщения	
	PName	Суммарная выручка	
1	Iphone 6	20000	
2	Iphone 7	50000	
3	Iphone 8	48000	

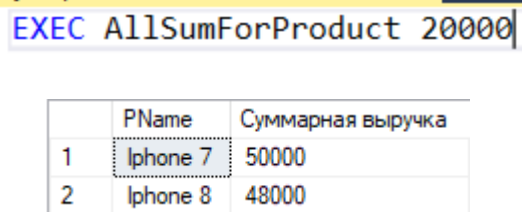
Рисунок 37 – результат выполнения AllSumForProduct

d) Улучшим предыдущую хранимую процедуру, чтобы в указанном списке остались только те товары, которые обеспечили заказы на сумму, превышающую N рублей (рисунок 38).

```
ALTER PROCEDURE AllSumForProduct
@MinRevenue FLOAT
AS
BEGIN
SELECT P.PName, SUM(Od.TotalPrice) AS 'Суммарная выручка'
FROM [ORDERDETAILS] Od
JOIN [PRODUCT] P ON Od.IdProduct = P.IdProduct
GROUP BY P.PName
HAVING SUM(Od.TotalPrice) > @MinRevenue
END;
```

Рисунок 38 – улучшенная хранимая процедура AllSumForProduct

Вызовем полученную функцию (Товары, которые обеспечили заказы на сумму, больше, чем 20000) (рисунок 39).



```
EXEC AllSumForProduct 20000
```

	PName	Суммарная выручка
1	Iphone 7	50000
2	Iphone 8	48000

Рисунок 39 – результат улучшенной хранимой процедуры (выводит заказы на сумму от 20000)

е) С учетом всех товаров и их габаритных размеров в составе одного заказа оцените, хотя бы примерно, минимальный объем заказа, если его аппроксимировать прямоугольным параллелепипедом, т.е рассчитайте габаритные размеры заказа: длину, ширину, высоту. Каждый из габаритных размеров заказа будет не менее суммы габаритных размеров, выстроенных друг за другом товаров. Товары в составе заказа можно вращать, ставить друг на друга, ставить вплотную друг с другом, но нельзя сжимать, разрезать и менять их геометрическую форму. Каждый товар считать прямоугольным параллелепипедом с заданной длиной, шириной и высотой. (рисунок 40).

```

ALTER TRIGGER MinimalVolume
ON ORDERDETAILS AFTER INSERT, UPDATE
AS
BEGIN
    DECLARE @Width FLOAT;
    DECLARE @Height FLOAT;
    DECLARE @Depth FLOAT;
    DECLARE @Price INT;
    DECLARE @Sizes Table(W FLOAT, H FLOAT, D FLOAT, Price INT);
    DECLARE @IdOrder INT;
    SELECT TOP 1 @IdOrder = IdOrder FROM inserted

    INSERT INTO @Sizes SELECT dbo.Maxxx(p.PWidtht, p.PHeight, p.PDepth) as w,
        dbo.Minnn(p.PWidtht, p.PHeight, p.PDepth) as h,
        dbo.Maxxx2(p.PWidtht, p.PHeight, p.PDepth) as d,
        oc.Count * oc.TotalPrice
        FROM Product as p INNER JOIN(SELECT *
        FROM ORDERDETAILS as od
        WHERE od.IdOrder = @IdOrder) as oc
        ON p.IdProduct = oc.IdProduct

    SELECT TOP 1 @Width = w FROM @Sizes ORDER BY w DESC
    SELECT TOP 1 @Height = h FROM @Sizes ORDER BY h DESC
    SELECT TOP 1 @Depth = d FROM @Sizes ORDER BY d DESC
    SELECT @Price = SUM(Price) from @Sizes

    UPDATE [dbo].[ORDER]
    SET BoxWidth = @Width,
        BoxHeight = @Height,
        BoxDepth = @Depth
    WHERE IdOrder IN (SELECT TOP 1 IdOrder FROM inserted)
END;

```

Рисунок 40 – хранимая процедура для расчета габаритных размеров

Заключение

В результате выполнения курсовой работы были достигнуты цели, поставленные во введении. Были сформированы общие навыки научно-исследовательской деятельности, создания и оформления научных работ.

Работая с научной, технической и справочной литературой, были углублены теоретические знания о ранних моделях данных, их структуре и применении.

При выполнении практической части работы были освоены базовые приемы работы с СУБД Microsoft SQL Server. Также были приобретены навыки проектирования реляционных баз данных, создания базы данных, применения декларативных языков манипулирования данными. Углублены знания о концептуальном моделировании, нормализации, физическом моделировании и программировании на языке SQL. Была создана база данных с таблицами, атрибутами, ограничениями, связями, представлениями, диаграммой, и хранимыми процедурами.

Список литературы

1. Агальцов, Виктор Петрович. Базы данных: учебник / В. П. Агальцов. Кн.2 : Распределенные и удаленные базы данных. - М.: Форум: ИНФРА-М, 2016. - 272 с.
2. Дейт К. Дж. Введение в системы баз данных. - 6-е изд.: Пер. с англ. - СПб.: Издательский дом «Вильямс», 2000.
3. Вилдермьюс Шон. Практическое использование ADO.NET, доступ к данным в Internet. — Москва: Вильямс, 2003. 288 с.
4. Когаловский М.Р. Энциклопедия технологий баз данных. - М.: Финансы и статистика, 2002.
5. Коннолли, Томас, Бегг, Каролин. Базы данных. Проектирование, реализация и сопровождение. Теория и практика. 3-е издание.: Пер. с англ. — М.: Издательский дом «Вильямс», 2017. — 1440 с., В. П. Агальцов. Распределенные и удаленные базы данных. - М.: Форум: ИНФРА-М, 2016.
6. Красина Ф.А. Базы данных: Учебное пособие. –Томск: Томский межвузовский центр дистанционного образования, 2002. -114с.
Медведкова, И.Е. / И.Е. Медведкова, Ю.В. Бугаев, С.В. Чикунов. - Воронеж: Воронежский государственный университет инженерных технологий, 2014 - 105 с.