МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«ЮЖНО-УРАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»

Факультет ***Электротехнический***

Кафедра ***автоматики***

Направление подготовки ***27.03.04 «Управление в технических системах»***

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой

_____

ГолощаповС. С._____

_____2021 г.

База данных информационной подсистемы отдела продаж торгово-промышленной компании

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ– 27.03.04.2021.172.00.00ПЗ ВКР

Автор ВКР
студент группы_МиЭт- 426_____
_____/Романчиков Т.Д_____
подпись ФИО
_____2021 г.

Руководитель работы
Старший преподаватель_____
должность
_____/___Казанцев А.М._____
подпись ФИО
_____2021 г.

Консультант

начальник отдела ООО ТПК «ЖилМебСтрой»__
должность
_____/Трусов С.К._____
подпись ФИО
_____2021 г.

Нормоконтроль

старший преподаватель_____
должность
_____/_____Елисеев В.П._____
подпись ФИО
_____2021 г.

Миасс 2021

АННОТАЦИЯ

Романчиков Т.Д. Выпускная квалификационная работа. База данных информационной подсистемы отдела продаж торгово-промышленной компании − Миасс: ЮУрГУ – 2021. − 76 с, 7 илл., 7 табл., библиогр. список – 17 наим., 2 прил.

В ходе написания работы была разработана база данных информационной подсистемы отдела продаж торгово-промышленной компании. Назначение базы данных – структуризация каталога документов, автоматизация поиска и редактирование данных, формирование контрольных отчетов о количестве, объеме продаж, работе сотрудников.

В отчет о проделанной работе входит разработанная база данных, которая в полной мере удовлетворяет требованиям предприятия к базе данных информационной подсистемы отдела продаж, учитывает основные направления деятельности отдела продаж, учитывает специфику его работы. При этом база данных крайне проста в применении, имеет удобный интерфейс работы.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Успешное развитие бизнеса в настоящее время во основном зависит от применения современных информационных технологий и внедрения новейших, обрабатывающих любую информацию с максимальной эффективностью, в связи с чем необходимо использование средств, выполняющих эффективную обработку, хранение, а также изменение и распределение накопленных данных.

Торгово-промышленная компания «ЖилМебСтрой» нуждается в электронной базе данных (БД) для работы отдела продаж с внутренними документами, т.к. стабильная работа отдела имеет не маловажное значение для оптимальной работы предприятия в совокупности.

Актуальность выбора базы данных в качестве темы ВКР обусловлена необходимостью проектирования и разработки базы данных информационной системы в связи со специфическими особенностями исследуемого предприятия с целью формирования наиболее эффективной базы данных.

Одна из составляющих этого процесса – формирование электронного каталога документации отдела продаж, который обеспечит структуризацию каталога документов, автоматизацию поиска и редактирование данных, формирование контрольных отчетов для проверки наличия документов, их движения.

Задачу составления каталога не стоит рассматривать только как некую программу, реализующую функцию механического распределения документации, необходимо, чтобы программа включала в себя средства для случая изменения некоторых входных данных.

Формирование электронных отчетов позволяет существенно сократить время на выборку данных, а также снижает ошибки в результате человеческого фактора, ввиду того, что все отчеты строятся автоматически.

Базы данных в современном мире составляют основу информационных процессов, существующих в различных сферах человеческой деятельности, и являются оптимальным средством представления данных, связей между ними и

их структур, а также управления ими. «Интегрированные средства хранения информации, обеспечивающие централизованное управление данными соответствуют концепции БД, при этом база данных должна поддерживаться в среде ЭВМ единым программным обеспечением - системой управления базами данных (СУБД).

Предметной областью можно назвать фрагмент реальности, описываемый или моделируемый с помощью базы данных и ее приложений. Выделяемые в предметной области информационные объекты – идентифицируемые объекты реального мира, процессы, системы, понятия и т.д., сведения о которых хранятся в базе данных».[17]

1 АНАЛИЗ ТЕКУЩЕГО СОСТОЯНИЯ, ПОСТАНОВКА ЗАДАЧИ

Работа с клиентами является основной функцией отдела продаж, заключается в работе с целевыми клиентами. Целевой сегмент представляет собой потребности, интересы и цены. Также необходима работа с поставщиками и каналами распределения товаров.

И немаловажным фактором является текущее состояние деятельности торгово-промышленной компании.

В работе информационной подсистемы отдела продаж ООО ТПК «ЖилМебСтрой» обнаружены следующие проблемы:

- отсутствует автоматизированный учет продаж, а также документации по количеству сделок и их суммы;
- реализация оформления документации и отчетов в бумажном виде;
- анализ отчетов начальником отдела осуществляется вручную;
- составление собственных отчетов начальником отдела для руководства предприятия производится также на бумажных носителях.

Анализ существующих проблем выявил необходимость в разработке базы данных информационной подсистемы в связи со специфическими особенностями исследуемого предприятия с целью формирования наиболее эффективной базы данных. Реализация данной задачи возможна в виде приложения.

Приложение должно выполнять следующие задачи:

- структуризацию каталога документов отдела продаж:
- автоматизацию поиска и редактирование данных в БД;
- оформление отчетов и выборку необходимых данных.

Выполняемые функции БД:

- ведение каталога данных сотрудников отдела продаж;
- ведение каталога данных клиентов;
- ведение каталога данных поставщиков;
- ведение каталога данных товаров;
- учёт продаж по заданным параметрам;

− формирование прайс-листа с учетом наличия товаров.

Каталоги, входящие в базу данных: товары, клиенты, сотрудники, поставщики, единицы измерения.

Для простоты визуализации и использования приложение должно иметь понятный интерфейс.

2 ОБОСНОВАНИЕ ВЫБОРА СРЕДЫ РЕАЛИЗАЦИИ

2.1 Сравнительный анализ СУБД

Различные БД предоставляют информацию в динамическом виде, выбор необходимо производить в зависимости от текущего состояния информационных технологий на предприятии, потребностей предприятия, а также с учетом бюджета.

База данных разрабатывается для производственного предприятия, занимающегося производством технологичной продукции и оптовой торговлей прочими строительными материалами и изделиями.

При рассмотрении известных СУБД можно сделать вывод, что почти во всех из них путем применения средств разработки приложений достигается автоматизированная обработка данных.

2.1.1 ORACLE

«Oracle RDBMS (она же Oracle Database) на первом месте среди СУБД. Система популярна у разработчиков, проста в использовании, у нее понятная документация, поддержка длинных наименований, JSON, улучшенный тег списка и Oracle Cloud.

- Разработчик: Oracle Corporation.
- Написана на:Assembly, C, C++.
- Блог: Oracle NoSQL.
- Последняя версия: 18.3Особенности:
- Обрабатывает большие данные.
- Поддерживает SQL, к нему можно получить доступ из реляционных БД Oracle.
- Oracle NoSQL Database с Java/C API для чтения и записи данных.» [16]

2.1.2 MySQL

Oracle MySQL Database Service - это полностью управляемая служба баз данных, которая позволяет разработчикам быстро разрабатывать и развертывать безопасные облачные приложения с использованием самой популярной в мире базы данных с открытым исходным кодом. Служба баз данных MySQL - единственная облачная служба MySQL со встроенным высокопроизводительным ускорителем запросов в памяти - HeatWave, который позволяет клиентам выполнять сложную аналитику непосредственно в отношении своих действующих баз данных MySQL, устраняя необходимость в сложных, трудоемких и трудоемких задачах. дорогостоящее перемещение данных и интеграция с отдельной аналитической базой данных. HeatWave на несколько порядков увеличивает производительность MySQL для аналитических и транзакционных запросов.

Сервер MySQL имеет следующие характеристики:

- Единая база данных для OLTP и OLAP;
- Встроенная аналитика в реальном времени;
- Единый унифицированный сервис для OLTP и OLAP;
- Устранение стоимости, сложности и риска ETL;
- Повышенная безопасность:
- Соответствие отраслевым и нормативным требованиям;
- Оптимизирован для Oracle Cloud Infrastructure (OCI);
- 400-кратное ускорение запросов MySQL;
- В 1100 раз быстрее, чем Amazon Aurora;
- В 2,7 раза быстрее, чем Amazon Redshift.

«Несмотря на то, что MySQL постоянно совершенствуется, он уже сегодня обеспечивает широкий спектр полезных функций. Благодаря своей доступности, скорости и безопасности MySQL очень хорошо подходит для доступа к базам данных по Internet.

- Разработчик: Oracle Corporation

- Написана на C, C++
- Последняя версия: 8.0.16

Особенности:

- Масштабируемость.
- Лёгкость использования.
- Безопасность.
- Поддержка Novell Cluster.
- Скорость.
- Поддержка многих операционных систем.»[16]

2.1.3 Microsoft SQL Server

SQL Server является одной из наиболее популярных систем управления базами данных (СУБД) в мире. «Данная СУБД подходит для самых различных проектов: от небольших приложений до больших высоконагруженных проектов. SQL Server долгое время был исключительно системой управления базами данных для Windows, однако начиная с версии 16 эта система доступна и на Linux.

Для взаимодействия с базой данных применяется язык SQL (Structured Query Language). Клиент (например, внешняя программа) отправляет запрос на языке SQL посредством специального API. СУБД должным образом интерпретирует и выполняет запрос, а затем посылает клиенту результат выполнения.» [15]

«И графический интерфейс, и программное обеспечение основаны на командах. Поддерживает SQL, непроцедурные, нечувствительные к регистру и общие языки баз данных.» [16]

Сильные стороны SQL обеспечивают преимущества для всех типов пользователей, включая программистов приложений, администраторов баз данных, менеджеров и конечных пользователей. С технической точки зрения SQL - это подъязык данных. Цель SQL - предоставить интерфейс для реляционной

базы данных, такой как Oracle Database, и все операторы SQL являются инструкциями для этой базы данных. В этом SQL отличается от языков программирования общего назначения, таких как C и BASIC. Среди особенностей SQL можно выделить следующие: Он обрабатывает наборы данных как группы, а не как отдельные единицы. Он обеспечивает автоматическую навигацию к данным. Он использует сложные и мощные по отдельности утверждения, которые, следовательно, автономны. Операторы управления потоком изначально не были частью SQL, но они находятся в необязательной части SQL, ISO / IEC 9075-4: 2011. Операторы управления потоком обычно известны как «постоянные хранимые модули» (PSM), а расширение PL / SQL для Oracle SQL аналогично PSM.

SQL позволяет работать с данными на логическом уровне. Все операторы SQL используют оптимизатор, часть Oracle Database, которая определяет наиболее эффективные способы доступа к указанным данным. Oracle также предоставляет методы, которые можно использовать для улучшения работы оптимизатора.

SQL предоставляет операторы для множества задач, в том числе: запрос данных, вставка, обновление и удаление строк в таблице, создание, замена, изменение и отбрасывание объектов, управление доступом к базе данных и ее объектам. Гарантия согласованности и целостности базы данных SQL объединяет все предыдущие задачи на одном согласованном языке.

2.1.4 PostgreSQL

«Масштабируемая объектно-реляционная база данных, работающая на Linux, Windows, OSX и некоторых других системах. В PostgreSQL 10 есть такие функции, как логическая репликация, декларативное разбиение таблиц, улучшенные параллельные запросы, более безопасная аутентификация по паролю на основе SCRAM-SHA-256.

- Разработчик: PostgreSQL Global Development Group

- Написана на C

- Используется в компаниях: Apple, Cisco, Fujitsu, Skype, and IMDb

Особенности:

- Поддержка табличных пространств, а также хранимых процедур, объединений, представлений и триггеров.

- Восстановление на момент времени (PITR).

- Асинхронная репликация.» [16]

2.1.5 MongoDB

«Самая популярная NoSQL система управления базами данных. Лучше всего подходит для динамических запросов и определения индексов. Гибкая структура, которую можно модифицировать и расширять. Поддерживает Linux, OSX и Windows, но размер БД ограничен 2,5 ГБ в 32-битных системах. Использует платформы хранения MMAPv1 и WiredTiger.

Разработчик: MongoDB Inc. в 2007

- Написана на C++

Особенности:

- Высокая производительность.
- Автоматическая фрагментация.
- Работа на нескольких серверах.
- Поддержка репликации Master-Slave.
- Данные хранятся в форме документов JSON.
- Возможность индексировать все поля в документе.
- Поддержка поиска по регулярным выражениям.» [16]

2.1.6 DB2

«Работает на Linux, UNIX, Windows и мейнфреймах. Эта СУБД идеально подходит для хост-сред IBM. Версию DB2 Express-C нельзя использовать в средах

высокой доступности (при репликации, кластеризации типа active-passive и при работе с синхронизируемым доступом к разделяемым данным).

- Разработчик: IBM
- Написана на C, C++, Assembly
- Последняя версия: 11.1

Особенности DB2 11.1:

- Улучшенное встроенное шифрование.
- Упрощённая установка и развёртывание.» [16]

2.1.7 Microsoft Access

СУБД от Microsoft, включающая графический интерфейс, ядро Microsoft Jet, а также имеющая инструменты разработки программного обеспечения, входит в набор программ Microsoft Office.

«Идеально подходит для начала работы с данными, но производительность не рассчитана на большие проекты. В MS Access можно использовать C, C#, C++, Java, VBA и Visual Rudimental.NET. Access хранит все таблицы БД, запросы, формы, отчёты, макросы и модули в базе данных Access Jet в виде одного файла.

- Разработчик: Microsoft Corporation
- Последняя версия: 16.0

Особенности:

- Можно использовать VBA для создания многофункциональных решений с расширенными возможностями управления данными и пользовательским контролем.
- Импорт и экспорт в форматы Excel, Outlook, ASCII, dBase, Paradox, FoxPro, SQL Server и Oracle.
- Формат базы данных Jet.» [16]

2.1.8 Cassandra

«СУБД активно используется в банковском деле, финансах, а также в Facebook и Twitter. Поддерживает Windows, Linux и OSX. Для запросов к БД Cassandra используется SQL-подобный язык — Cassandra Query Language (CQL).

- Разработчик: Apache Software Foundation
- Написана на: Java
- Последняя версия: 3.11.4

Особенности:

- Линейная масштабируемость.
- Быстрое время отклика.
- Поддержка MapReduce и Apache Hadoop.
- Максимальная гибкость.
- P2P архитектура.»[16]

2.1.9 Redis

«Redis или Remote Dictionary Server — СУБД с открытым исходным кодом, которая снабжена механизмами журналирования и снимков. Поддерживаются списки, строки, хэши, наборы. Используется для БД, брокеров сообщений и кэшей. Все операции в Redis атомарные. Система написана на языке C и поддерживается практически всеми языками программирования.

- Разработчик: Salvatore Sanfilippo
- Последняя версия: 5.0.5

Особенности:

- Автоматическая обработка отказа.
- Транзакции.
- Сценарии LUA.
- Вытеснение LRU-ключей.
- Поддержка Publish/Subscribe.» [16]

2.1.10 Elasticsearch

«Легко масштабируемая поисковая система корпоративного уровня с открытым исходным кодом. Благодаря обширному и продуманному API обеспечивает чрезвычайно быстрый

поиск, работает в том числе с приложениями для обнаружения данных. Используется такими компаниями, как Википедия, The Guardian, StackOverflow, GitHub. ElasticSearch позволяет создавать копии индексов и сегментов.

- Разработчик: Elastic NV
- Написана на Java
- Последняя версия: 7.2.0

Особенности:

- Масштабируемость вплоть до нескольких петабайт структурированных и неструктурированных данных.
- Многопользовательская поддержка.
- Масштабируемый поиск, поиск в режиме реального времени.» [16]

2.2 Обоснование выбора среды реализации приложения

Для управления данными выбрана СУБД MS SQL Server 2008 Express Edition. Выбор данной СУБД обусловлен следующим:

− Включенные в SQL Server библиотеки Declarative Management Framework повышают эффективность администрирования, а также позволяют распределять полномочия для баз или существующих отдельных таблиц. Методы компрессии данных, по сравнению с предыдущими версиями, улучшены.

− Благодаря встроенной поддержке .NET Framework хранимые процедуры БД можно написать на любом языке платформы .NET.

Microsoft SQL Server может быть бесплатно распространяемой версией с небольшими техническими ограничения либо коммерческой версией SQL Server с расширенными возможностями. Бесплатная версия Microsoft SQL Server Express

Edition ограничивает применение СУБД для ведения программных задач в рамках небольшой компании. Microsoft SQL Server содержит полную поддержку типов данных, включая XML–спецификации, а также национальных алфавитов и стандарт кодирования символов Unicode.

Для разработки приложения был выбран язык программирования C# и среда разработки Visual Studio 2008.

2.2.1 Начало работы с MS SQL Server

Под базой данных в основном понимают только наличие таблиц с данными, но в ней, помимо таблиц, содержатся представления, функции, хранимые процедуры.

При создании собственной базы данных можно вложить и хранить любые данные, но для работы СУБД необходимы системные БД. В таблице 2.1 представлены системные базы данных SQL Server.

Таблица 2.1 – Системные базы данных SQL Server

Системные базы данных	
Системная база данных	Description
База данных master	В этой базе данных хранятся все данные системного уровня для экземпляра SQL Server.
База данных msdb	Используется агентом SQL Server для планирования предупреждений и задач.
Шаблон базы данных	Используется в качестве шаблона для всех баз данных, создаваемых в экземпляре SQL Server. Изменение размера, параметров сортировки, модели восстановления и других параметров базы данных model приводит к изменению соответствующих параметров всех баз данных, создаваемых после изменения.
База данных ресурсов	База данных только для чтения. Содержит системные объекты, которые входят в состав SQL Server. Системные объекты физически хранятся в базе данных Resource , но логически отображаются в схеме sys любой базы данных.
База данных tempdb	Рабочее пространство для временных объектов или взаимодействия результирующих наборов.

На рисунке 2.1 показано отображение системных БД в узле Databases -> System Databases:

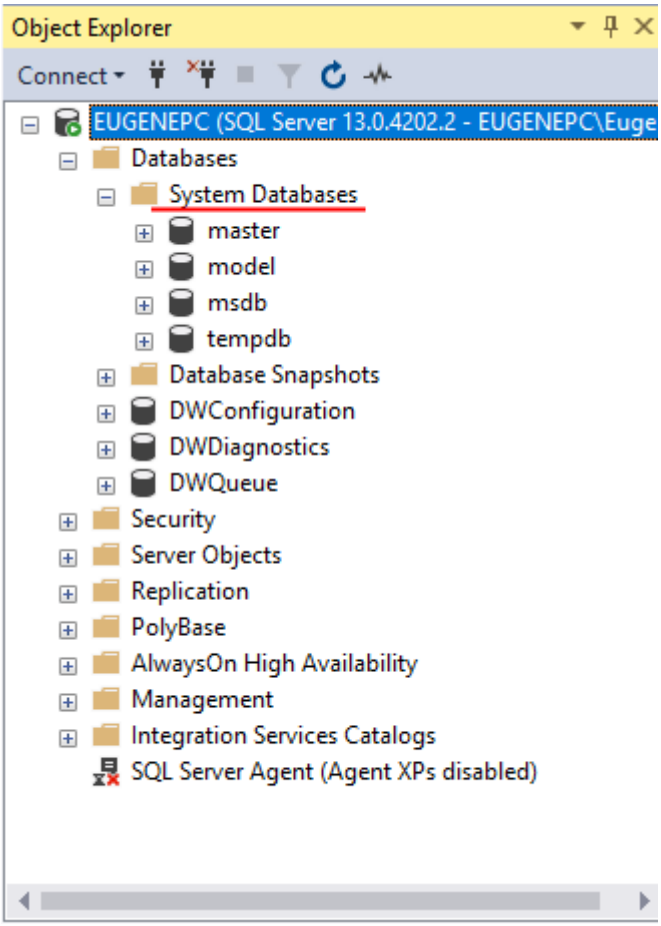

Рисунок 2.1 – Отображение системных баз в списке

Системные базы данных не подлежат изменению.

На экземпляре SQL Server можно указать максимум 32 767 баз данных.

Для создания собственной БД используется команда CREATE DATABASE. Оператор CREATE DATABASE должен выполняться в режиме автоматической фиксации (режим управления транзакциями по умолчанию) и не допускается в явной или неявной транзакции.

Требуется разрешение CREATE DATABASE в базе данных master или требуется разрешение CREATE ANY DATABASE или ALTER ANY DATABASE. Чтобы сохранить контроль над использованием диска в экземпляре SQL Server, разрешение на создание баз данных обычно ограничивается несколькими учетными записями.

После создания БД, нужно ее установить с применением команды USE:

```
USE usersdb
```

Если существует файл, представляющий собой файл с расширением mdf, этот файл можно переносить. Однако, даже если скопировать его на компьютер с установленной СУБД, появления скопированной БД на сервере не будет, требуется прикрепление БД к серверу.

Команды прикрепления базы данных:

```
CREATE DATABASE название_базы_данных
ON PRIMARY(FILENAME='путь_к_файлу_mdf_на_локальном_компьютере')
FOR ATTACH;
```

Указание места хранения может быть произвольным, но лучше остановить выбор на каталоге, хранящем БД сервера. На Windows 10 по умолчанию это каталог

C:\ProgramFiles\MicrosoftSQLServer\MSSQL13.MSSQLSERVER\MSSQL\DATA.

Для удаления собственной БД используется команда DROP DATABASE, имеющая синтаксис:

```
DROP DATABASE database_name1 [, database_name2]...
```

Удалить все необходимые для этого БД одновременно можно путем перечисления после записи команды через запятую.

Необходимо помнить, что даже если удаляемая БД была прикреплена, то все равно будут удалены все файлы.

База данных master должна быть скопирована всякий раз, когда пользовательская база данных создается, изменяется или удаляется.

Для создания таблиц применяется команда CREATE TABLE. С этой командой используется ряд операторов, которые определяют столбцы таблицы и их атрибуты. И кроме того, можно использовать ряд операторов, которые определяют свойства таблицы в целом. Одна база данных может содержать до 2 миллиардов таблиц.

Общий синтаксис создания таблицы выглядит следующим образом:

```
CREATE TABLE название_таблицы
(название_столбца1 тип_данных атрибуты_столбца1,
 название_столбца2 тип_данных атрибуты_столбца2,
 ..........................................
```

```
 название_столбцаN тип_данных атрибуты_столбцаN,
 атрибуты_таблицы
)
```

После команды CREATE TABLE идет название создаваемой таблицы. Имя таблицы выполняет роль ее идентификатора в БД. Имя должно иметь длину не больше 128 символов. Имя может состоять из алфавитно-цифровых символов, а также символов $ и знака подчеркивания. Причем первым символом должна быть буква или знак подчеркивания.

Имя объекта не может включать пробелы и не может представлять одно из ключевых слов языка Transact-SQL. Если идентификатор все же содержит пробельные символы, то его следует заключать в кавычки. Если необходимо в качестве имени использовать ключевые слова, то эти слова помещаются в квадратные скобки.

Примеры корректных идентификаторов:

```
Users
tags$345
users_accounts
"users accounts"
[Table]
```

После имени таблицы в скобках указываются параметры всех столбцов и в самом конце атрибуты, которые относятся ко всей таблице. Атрибуты столбцов и атрибуты таблицы являются необязательными компонентами, и их можно не указывать.

В самом просто виде команда CREATE TABLE должна содержать как минимум имя таблицы, имена и типы столбцов.

Таблица может содержать от 1 до 1024 столбцов. Каждый столбец должен иметь уникальное в рамках текущей таблицы имя, и ему должен быть назначен тип данных.

Столбцы с типом INT будут хранить числовые значения, столбцы с типом NVARCHAR(20) представляют строку UNICODE длиной не более 20 символов, с типом VARCHAR(30/20) - хранят строку, но не в кодировке UNICODE.

Для удаления таблиц используется команда DROP TABLE, которая имеет следующий синтаксис:

```
DROP TABLE table1 [, table2, ...]
```

Для переименования таблиц применяется системная хранимая процедура "sp_rename". Например, переименование таблицы Users в UserAccounts в базе данных usersdb:

```
USE usersdb;
EXEC sp_rename 'Users', 'UserAccounts';
```

При создании таблицы для всех ее столбцов необходимо указать определенный тип данных.

Типы данных и функции даты и времени TSQL:

@@ DATEFIRST - возвращает первый день каждой недели. CURRENT_TIMESTAMP - возвращает текущую системную временную метку базы данных.

DATEADD - возвращает дату с добавленным интервалом указанного числа. DATEDIFF - возвращает разницу между указанной начальной и конечной датами. DATEFROMPARTS - возвращает значение даты для указанного года, месяца или дня.

DATENAME - возвращает строку, представляющую указанную часть указанной даты.

DATEPART - возвращает целое число, представляющее указанную часть указанной даты.

DATETIMEFROMPARTS - возвращает значение datetime для указанных аргументов даты и времени.

DATETIME2FROMPARTS - возвращает значение datetime2 для указанных аргументов даты и времени.

DAY - возвращает с даты целое число, представляющее день месяца. EOMONTH - возвращает последний день месяца.

GETDATE () - возвращает текущую системную временную метку базы данных.

GETUTCDATE () - возвращает текущую системную временную метку базы данных.

ISDATE - возвращает 1 для допустимой даты, времени или значения datetime. в противном случае 0.

MONTH - возвращает значение типа int, представляющее месяц указанной даты.

SMALLDATETIMEFROMPARTS - возвращает значение smalldatetime для указанной даты и времени.

SWITCHOFFSET - возвращает значение datetimeoffset, измененное относительно сохраненного смещения часового пояса.

SYSDATETIME - возвращает значение datetime2 (7), которое содержит дату и время компьютера, на котором запущен экземпляр SQL Server. SYSDATETIMEOFFSET - возвращает значение datetimeoffset (7), которое содержит дату и время компьютера, на котором запущен экземпляр SQL Server. SYSUTCDATETIME - дата и время возвращаются в формате UTC (всемирное координированное время).

TIMEFROMPARTS - возвращает значение времени для указанного времени и с указанной точностью.

TODATETIMEOFFSET - возвращает значение datetimeoffset, преобразованное из выражения datetime2.

YEAR - возвращает значение типа int, представляющее год указанной даты.

Язык Transact SQL позволяет использовать различные типы данных, такие как: числовые (int, numeric, decimal, float), символьные строки (char, varchar), символьные строки Unicode (nchar, nvarchar), Date (date, datetime, datetime2, time) и другие типы данных.

Двоичные строки

Binary - двоичные данные фиксированной длины.

Varbinary - двоичные данные переменной длины.

Строки символов

Char - строковые данные фиксированного размера.

Varchar - строковые данные переменного размера.

Числовые

Bigint - определяет целочисленный тип данных с хранилищем 8 байт.

Int - основной целочисленный тип данных в SQL Server, имеет хранилище 4 байта.

Smallint - определяет целочисленный тип данных с хранилищем 2 байта.

Tinyint - определяет целочисленный тип данных с хранением в 1 байт.

Decimal - определяет числовой тип данных с фиксированной точностью и масштабными числами.

Numeric - функционально идентичен десятичному.

Bit - целочисленный тип данных, который может принимать значение 1, 0 или NULL.

Float - определяет приблизительные числовые значения.

Real - определяет приблизительные числовые значения с памятью 4 байта.

Строки символов Юникода

Nchar - строковые данные Unicode фиксированного размера.

Nvarchar - строковые данные Unicode переменного размера.

Другие типы данных

Rowversion - предоставляет автоматически генерируемые уникальные двоичные числа.

Uniqueidentifier - считается символьным типом для целей преобразования.

Table - сохранить набор результатов для обработки в более позднее время.

Cursor - определяет тип данных курсора для переменных или выходных параметров хранимой процедуры.

SQL_VARIANT - хранит значения различных SQL Server.

XML - определяет тип данных xml.

T-SQL поддерживает функциональность подзапросов (subquery), то есть таких запросов, которые могут встроены в другие запросы.

Агрегатные функции TSQL:

AVG - возвращает среднее значение.

COUNT - возвращает количество строк.

MAX - возвращает максимальное значение.

MIN - возвращает минимальное значение.

СУММ - возвращает сумму всех значений.

Строковые функции TSQL:

Charindex - возвращает начальную позицию.

Concat - возвращает строку в результате конкатенации.

Left - возвращает левую часть символьной строки.

Len - возвращает количество символов строкового выражения без конечных пробелов.

Lower - возвращает символьное выражение в нижнем регистре.

Ltrim - возвращает символьное выражение после удаления ведущих пробелов.

Substring - возвращает часть символа.

Patindex - возвращает начальную позицию первого вхождения шаблона в указанное выражение.

Replace - заменить значения указанной строки на значения другой строки.

Right - возвращает правую часть символьной строки.

Rtrim - возвращает символьную строку после усечения всех конечных пробелов.

Upper - возвращает символьное выражение в верхнем регистре.

Системные функции TSQL:

@@ CONNECTIONS - возвращает количество попыток подключения.

@@ ERROR - возвращает номер ошибки для последнего выполненного оператора Transact-SQL.

@@ IDENTITY - возвращает последнее вставленное значение идентичности.

@@ ROWCOUNT - возвращает количество строк, затронутых последним оператором.

COALESCE - возвращает тип данных выражения с наивысшим приоритетом типа данных.

ERROR_LINE - возвращает номер строки появления ошибки.

ERROR_MESSAGE - возвращает текст сообщения об ошибке.

ERROR_NUMBER - возвращает номер ошибки.

ERROR_PROCEDURE - возвращает имя хранимой процедуры или триггера, в котором возникла ошибка.

ERROR_SEVERITY - возвращает значение серьезности ошибки, в которой возникла ошибка.

ERROR_STATE - возвращает номер состояния вызвавшей ошибки.

HOST_ID - возвращает идентификационный номер рабочей станции.

HOST_NAME - возвращает имя рабочей станции.

ISNULL - заменяет NULL указанным значением замены.

ISNUMERIC - определяет, является ли выражение допустимым числовым типом.

NULLIF - возвращает нулевое значение, если два указанных выражения равны.

Функции безопасности TSQL:

CURRENT_USER - возвращает имя текущего пользователя.

ORIGINAL_LOGIN - возвращает имя логина, подключенного к экземпляру SQL Server.

SESSION_USER - возвращает имя пользователя текущего сеанса в текущей базе данных.

SYSTEM_USER - возвращает имя пользователя для входа в систему.

USER_NAME - возвращает имя пользователя базы данных по указанному идентификационному номеру.

Функции метаданных:

TSQL APP_NAME - возвращает имя приложения для текущего сеанса.

DB_ID - возвращает номер id базы данных.

DB_NAME - возвращает имя базы данных.

OBJECT_DEFINITION - возвращает определение указанного объекта.

OBJECT_ID - возвращает идентификатор объекта базы данных.

OBJECT_NAME - возвращает имя объекта базы данных.

OBJECT_SCHEMA_NAME - возвращает имя схемы базы данных.

SCHEMA_ID - возвращает идентификатор схемы, связанный с именем схемы.

SCHEMA_NAME - возвращает имя схемы, связанное с идентификатором схемы.

Функции конфигурации TSQL:

@@ LOCK_TIMEOUT - возвращает текущую настройку тайм-аута блокировки в миллисекундах для текущего сеанса.

@@ MAX_CONNECTIONS - возвращает максимальное количество одновременных пользователей подключения разрешены на экземпляре SQL Server.

@@ SERVERNAME - возвращает имя локального сервера.

@@ SERVICENAME - возвращает имя раздела реестра, под которым работает SQL Server.

@@ SPID - возвращает идентификатор сеанса текущего пользовательского процесса.

Функции преобразования:

TSQL CAST - преобразовать выражение одного типа данных в другой.

CONVERT - преобразовать выражение одного типа данных в другой.

PARSE - возвращает результат выражения, преобразованный в запрошенный тип данных в SQL Server.

TRY_CAST - возвращает приведение значения к указанному типу данных, если приведение прошло успешно.

TRY_CONVERT - возвращает приведение значения к указанному типу данных, если приведение выполнено успешно.

TRY_PARSE - используется для преобразования выражений из строковых в типы даты / времени и числа.

Операторы T-SQL:

Арифметические операторы - сложение, вычитание, умножение, деление.

Побитовые операторы - and, or, исключающее or.

Операторы сравнения - равно, больше, меньше, не равно.

Составные операторы - сложение равно, вычитание равно, умножение равно.

Логические операторы - все и любые, между, существует, в, вроде, не или в некоторых.

Выборочный запрос T-SQL

Group By - GROUP BY используется, когда в запросе выбора существует агрегатная функция.

Having - определяет условие поиска для группы или агрегатной функции.

Order By- сортировка строк, возвращаемых в запросе на выборку.

Where - указывает условие поиска для строк, возвращаемых запросом на выборку.

Between - возвращает строки, если значение находится в пределах диапазона сравнений.

Like - возвращает строки, если символьная строка соответствует шаблону.

Exists - вернуть строки, если подзапрос содержит какие-либо строки.

In - возвращает строки, если указанное значение соответствует любому значению в подзапросе.

Some - сравнивает значение с набором значений в одном столбце.

Any - возвращает строки, если какое-либо значение равно одному значению в подзапросе.

Соединения таблиц T-SQL

Внутреннее соединение - возвращает строки из двух таблиц при обнаружении совпадения.

Left Join - возвращает все строки из левой таблицы, даже если нет совпадений с правой таблицей.

Right Join - возвращает все строки из правой таблицы, даже если нет совпадений с левой таблицей.

Self Join - используется для присоединения таблицы к самой себе.

Системные хранимые процедуры

Sp_addextendedproperty - добавляет новое расширенное свойство к объекту базы данных.

Sp_autostats - отображает или изменяет параметр автоматического обновления статистики.

Sp_columns - возвращает информацию о столбцах для указанных объектов базы данных.

Sp_column_privileges - возвращает информацию о привилегиях столбца для таблицы.

Sp_special_columns - возвращает оптимальный набор столбцов, однозначно идентифицирующих строку в таблице.

Sp_configure - отображает или изменяет глобальные параметры конфигурации для текущего сервера.

Sp_databases - показывает имя и размер всех баз данных для экземпляра SQL Server.

Sp_execute - выполняет подготовленную инструкцию Transact-SQL, используя указанный дескриптор и необязательное значение параметра.

Sp_executesql - выполняет инструкцию Transact-SQL, которую можно многократно использовать повторно.

Sp_fkeys - возвращает информацию о внешнем ключе.

Sp_help - выводит информацию об объектах базы данных.

Sp_helpconstraint - возвращает список всех типов ограничений.

Sp_helpdb - выводит информацию о базах данных.

Sp_helpindex - возвращает информацию об индексах таблицы или представления.

Sp_lock - показывает информацию о блокировках.

Sp_monitor - показывает статистику о Microsoft SQL Server.

Sp_prepare - подготавливает параметризованный оператор Transact-SQL и возвращает дескриптор оператора для выполнения.

Sp_pkeys - возвращает информацию о первичном ключе.

Sp_rename - изменяет имя объекта в текущей базе данных.

Sp_renamedb - изменяет имя базы данных.

Sp_tables - возвращает список объектов, таких как таблицы или представления.

Sp_helptrigger - возвращает тип или типы триггеров DML, определенных в указанной таблице.

Sp_table_privileges - возвращает список разрешений для указанной таблицы.

Sp_server_info - возвращает список имен атрибутов и соответствующих значений для SQL Server.

Sp_statistics - возвращает список всех индексов и статистики по указанной таблице.

Sp_stored_procedures - возвращает список хранимых процедур.

Sp_unprepare - отклоняет план выполнения, созданный хранимой процедурой sp_prepare.

Sp_updatestats - выполняет обновление статистики для всех пользовательских и внутренних таблиц в текущей базе данных.

Sp_who - возвращает информацию о текущих пользователях, сеансах и процессах в экземпляре SQL Server.

Резервное копирование всей базы данных выполняется с помощью BACKUP DATABASE. Синтаксис копирования:

```
    BACKUP DATABASE { database_name }
TO backup_device [ ,...n ]
[ MIRROR TO clause ]
[ WITH { DIFFERENTIAL | [ ,...n ] } ];
```

В следующем примере выполняется резервное копирование основной базы данных в файл на диске.

```
    BACKUP DATABASE master
TO DISK = 'E:\SQLServerBackup\master.bak'
```

```
WITH FORMAT;
GO
```

Выводимые сообщения:

Processed 464 pages for database 'master', file 'master' on file 1.

Processed 3 pages for database 'master', file 'mastlog' on file 1.

BACKUP DATABASE successfully processed 467 pages in 0.274 seconds (13.308 MB/sec).

Резервное копирование частичной базы данных также выполняется с помощью BACKUP DATABASE.

Используемый синтаксис:

```
BACKUP DATABASE { database_name }
READ_WRITE_FILEGROUPS [ , [ ,...n ] ]
TO backup_device [ ,...n ]
[ MIRROR TO clause ]
[ WITH { DIFFERENTIAL | [ ,...n ] } ];
```

В следующем примере частичное резервное копирование базы данных модели в файл на диске.

```
BACKUP DATABASE model READ_WRITE_FILEGROUPS
TO DISK = 'E:\SQLServerBackup\model_partial.bak'
GO
```

Выводимые сообщения:

Processed 328 pages for database 'model', file 'modeldev' on file 1.

Processed 2 pages for database 'model', file 'modellog' on file 1.

BACKUP DATABASE...FILE=name successfully processed 330 pages in 0.325 seconds (7.910 MB/sec).

Восстановление всей базы данных выполняется с помощью RESTORE DATABASE.

Syntax:

```
RESTORE DATABASE { database_name }
[ FROM [ ,...n ] ]
[ WITH
{
[ RECOVERY | NORECOVERY | STANDBY = {standby_file_name} ]
| , [ ,...n ]
```

```
| ,
| ,
| ,
| ,
| ,
} [ ,...n ]
]
[;]
```

Пример восстановления полной резервной копии базы данных с логического устройства резервного копирования.

```
      USE master;
GO
RESTORE DATABASE test_2
FROM test_2;
GO
```

Выводимые сообщения:

Processed 328 pages for database 'test_2', file 'test_2' on file 1.

Processed 2 pages for database 'test_2', file 'test_2_log' on file 1.

RESTORE DATABASE successfully processed 330 pages in 0.276 seconds (9.333 MB/sec).

Первичный ключ состоит из столбца или комбинации нескольких столбцов. Столбцы первичного ключа содержат уникальные значения, которые идентифицируют каждую строку в таблице. Максимальное количество столбцов, разрешенных для составления первичного ключа, составляет 16 столбцов, а общая длина ключа составляет 900 байт. Таблица может содержать только одно ограничение первичного ключа. Ограничения PRIMARY KEY не допускают значение NULL.

Внешний ключ - это ограничение - это связь между двумя таблицами, используемая для обеспечения целостности данных. Ограничения внешнего ключа могут ссылаться только на таблицы в одной базе данных на одном сервере.

Операторы транзакций T-SQL

Транзакция - это единая логическая единица работы, состоящая из нескольких операторов sql server. Транзакция начинается с выполнения первого

оператора sql server и заканчивается, когда транзакция сохраняется или откатывается. Операторы транзакций TSQL:

BEGIN DISTRIBUTED TRANSACTION - распределенная транзакция - это операционная транзакция, которая может выполняться на нескольких разных серверах.

BEGIN TRANSACTION - оператор Begin Transaction является начальной точкой явной транзакции и увеличивает @@ TRANCOUNT на 1.

COMMIT TRANSACTION - транзакция COMMIT - это конечная точка успешной неявной или явной транзакции. Когда @@ TRANCOUNT равно 1, все выполненные изменения данных фиксируются, и ресурсы, удерживаемые транзакцией, освобождаются и уменьшает @@ TRANCOUNT до 0. Когда @@ TRANCOUNT больше 1, COMMIT TRANSACTION уменьшает @@ TRANCOUNT только на 1 и транзакция остается активной.

COMMIT WORK - транзакция COMMIT WORK - это конечная точка успешной транзакции.

ROLLBACK TRANSACTION - транзакция ROLLBACK - это операция, которая откатывает неудачную явную или неявную транзакцию до начала транзакции или до точки сохранения внутри транзакции.

ROLLBACK WORK - это операция транзакции, которая откатывает указанную пользователем транзакцию.

SAVE TRANSACTION - транзакция SAVE устанавливает точку сохранения внутри транзакции.

На языке Transact sql операторы SET позволяют изменить обработку в текущем сеансе конкретной информации, например: формат даты, системный язык, тайм-аут блокировки, количество строк.

Дата и время:

SET Datefirst - устанавливает первый день недели на число от 1 до 7.

```
SET DATEFIRST { number | @number_variable } ;
```

SET Dateformat - устанавливает порядок частей даты месяца, дня и года.

```
SET DATEFORMAT { format | @format_variable } ;
```

Блокировка:

SET Deadlock_priority - устанавливает важность текущего сеанса, если он зашел в тупик с другим сеансом.

```
SET DEADLOCK_PRIORITY { LOW | NORMAL | HIGH | <numeric-priority> | @deadlock_variable } ;
<numeric-priority> ::= { -10 | -9 | ... | 0 | ... | 9 | 10 }
```

SET Lock_timeout - устанавливает количество миллисекунд оператора, ожидающего снятия блокировки.

```
SET LOCK_TIMEOUT milliseconds_number ;
```

Другие:

SET Concat_null_yields_null - проверяет, обрабатываются ли результаты конкатенации как нулевые или пустые строковые значения.

```
SET CONCAT_NULL_YIELDS_NULL { ON | OFF } ;
```

SET Cursor_close_on_commit - значение по умолчанию для CURSOR_CLOSE_ON_COMMIT - ВЫКЛ. Если для параметра CURSOR_CLOSE_ON_COMMIT установлено значение OFF, сервер не будет закрывать курсоры при фиксации транзакции.

```
SET CURSOR_CLOSE_ON_COMMIT { ON | OFF } ;
```

SET Identity_insert - разрешить вставку явных значений в столбец идентификаторов таблицы. Оператор IDENTITY_INSERT должен быть установлен в ON, чтобы вставить явное значение для столбца идентификаторов.

```
SET IDENTITY_INSERT [ database_name . [ schema_name ] . ] table { ON | OFF }
;
```

SET Language - устанавливает язык сеанса. Язык сеанса устанавливает формат даты и системных сообщений.

```
SET LANGUAGE { [ N ] 'language' | @language_variable } ;
```

Выполнение запроса:

SET Rowcount - устанавливает количество строк для sql-запроса. Когда возвращается указанное количество строк, выполнение запроса останавливается.

```
SET ROWCOUNT { number | @number_variable } ;
```

SET Noexec - устанавливает компиляцию каждого запроса, но не выполняет запросы.

```
SET NOEXEC { ON | OFF } ;
```

Подзапрос SQL (также называемый внутренним запросом или внутренним выбором) - это запрос sql, вложенный внутри оператора (SELECT, INSERT, UPDATE или DELETE) или внутри другого подзапроса.

В операторах, которые включают подзапрос SQL, обычно используются логические операторы (in, exists, all, any или some):

WHERE expression [NOT] IN (subquery)

WHERE expression comparison_operator [ANY | ALL | SOME] (subquery)

WHERE [NOT] EXISTS (subquery)

Операторы T-SQL - DML, DDL, DCL и TCL DDL (язык определения данных).

Операторы языка определения данных (DDL) определяют объекты в базе данных. Использовать операторы DDL можно для создания, изменения или удаления объектов в базе данных.

CREATE - оператор, используемый для начала создания новых объектов, таких как: база данных, таблица, представление, индекс, временная таблица, триггер, функция или процедура.

ALTER - оператор, используемый для изменения имени таблицы или столбцов таблицы (добавление, удаление, переименование), просмотра, триггера.

DROP - используется для удаления объектов в базе данных.

RENAME - используется для изменения имени объекта в базе данных.

DISABLE TRIGGER - отключает триггер состояния.

ENABLE TRIGGER - устанавливает активный триггер состояния.

COLLATIONS - определяет сортировку столбца базы данных или таблицы.

UPDATE STATISTICS - обновляет статистику оптимизации запросов для таблицы или индексированного представления.

DML (язык манипулирования данными).

Язык DML влияет на информацию, хранящуюся в базе данных. Использовать операторы DML можно для вставки, обновления и удаления строк в базе данных.

INSERT - вставить записи в таблицу.

DELETE - удалить записи в таблицу.

MERGE - вставка, обновление или удаление записей в таблице из результатов объединения.

BULK INSERT - импортирует записи из файла данных в таблицу или представление базы данных.

TRUNCATE TABLE - удалить все строки из таблицы.

DCL (язык управления данными) - утверждения разрешений. Язык управления данными (DCL) - операторы разрешений определяют, какие пользователи и логины могут получать доступ к данным и выполнять операции.

GRANT - инструкция, используемая для предоставления привилегий, ролей или разрешений на объекты базы данных.

REVOKE - используется для отзыва предоставленных разрешений.

Локальная переменная Transact-SQL - это объект базы данных, который может хранить одно значение данных определенного типа. Чтобы объявить переменную, используется ключевое слово DECLARE, присваивается имя переменной и тип данных.

DECLARE @MyVariable datatype;

После объявления переменной она получает значение NULL по умолчанию. Чтобы присвоить значение переменной, нужно использовать инструкцию SET.

3 РЕАЛИЗАЦИЯ МОДЕЛИ БД

3.1 Описание сущностей разрабатываемой модели базы данных и их атрибутов

Инфологическая модель – иначе это концептуальная модель – та модель предметной области, которая предназначена для представления семантики рассматриваемой предметной области на самом высоком уровне абстракции.

Сущность – другими словами это объектное множество, таблица, которые являются абстракцией происходящего процесса или явления, существующего объекта, о котором нужно хранить информацию.

Атрибут – признак сущности, который представляет собой наименьшую единицу структуры данных.

Атрибут должен иметь уникальное наименование для конкретного типа сущности.

Таблица 3.1 - Описание сущности концептуальной модели разрабатываемой БД

Наименование сущности	Описание сущности
Виды товаров	Каталог, содержащий информацию о видах товаров, реализуемых предприятием (классификация товаров)
Номенклатура товаров	Каталог, содержащий сведения о товарах
Товары в наличии	Список товаров, имеющихся в наличии, с указанием их количества
Список проданных товаров	Список товаров, занесенных в журнал продаж
Журнал продаж	Журнал, содержащий информацию о продажах
Единицы измерения	Список единиц измерения товаров, реализуемых предприятием
Поставщики	Каталог, содержащий информацию о поставщиках товаров
Сотрудники	Каталог, содержащий информацию о сотрудниках отдела продаж

Описание атрибутов сущностей представлено в таблице 3.2.

Таблица 3.2 - Описание заданных атрибутов

Название сущности	Атрибут	Описание атрибута
Поставщики	Наименование поставщика	Название организации
	ИНН поставщика	ИНН организации
Виды товаров	Идентификатор вида товаров	Уникальный номер вида товаров
	Классификация товаров	Название вида товаров
	Описание товара	Краткое описание товара
Номенклатура товаров	Идентификатор товара	Идентификатор, уникальный номер товара в номенклатуре
	Цена	Цена, по которой реализуется товар
Сотрудники	Идентификатор сотрудника	Уникальный номер сотрудника отдела продаж предприятия
	Фамилия, имя, отчество	Фамилия, имя, отчество сотрудника отдела продаж
	Должность	Должность сотрудника отдела продаж
Товары в наличии	Идентификатор записи	Уникальный номер записи о имеющихся в наличии товарах
	Идентификатор товара	Ссылка на товар
	Количество товаров	Количество товаров в наличии
Журнал продаж	Идентификатор записи	Уникальный номер записи о продажах
	Дата продажи	Дата совершения операции продажи, дата записи
	Покупатель (организация)	Название организации, купившей товары
	Покупатель (фамилия)	Фамилия покупателя
	Идентификатор сотрудника	Ссылка на сотрудника, оформившего продажу
Список проданных товаров	Идентификатор записи	Уникальный номер записи о проданном товаре
	Идентификатор товара	Ссылка на товар
	Количество	Количество единиц проданного товара
	Цена	Цена, по которой был продан указанный товар
Единицы измерения	Название единицы измерения	Название единицы измерения

3.2 Взаимосвязи таблиц

Модель «сущность-связь» можно составить используя три конструктивных элемента:

- сущность;
- атрибут;
- связь.

Типы взаимосвязей между таблицами разрабатываемой базы данных можно представить в следующем виде:

1. (1:1) - связь «один к одному»;

2. (1:M) - связь «один ко многим»;

3. (M:1) - связь «многие к одному»;

4. (M:N) - связь «многие ко многим».

На практике последний тип взаимосвязей применяется крайне редко, чтобы избежать потерь в производительности вычислений.

В разрабатываемой модели БД реализованы связи типа 2.

Описание взаимосвязей представлено в таблице 3.3.

Таблица 3.3 – Тип взаимосвязей между создаваемыми таблицами

Номер связи	Родительская таблица	Дочерняя таблица	Тип связи
1	Сотрудники	Журнал продаж	1:M
2	Журнал продаж	Список проданных товаров	1:M
3	Номенклатура товаров	Список проданных товаров	1:M
4	Виды товаров	Номенклатура товаров	1:M
5	Номенклатура товаров	Товары в наличии	1:M
6	Единицы измерения	Номенклатура товаров	1:M

4 ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ

4.1 Создание таблиц

Таблица 4.1 – Описание типов данных для таблиц и атрибутов в БД

Название таблицы	Название атрибута	Тип данных
contragent (поставщики)	contragent_id	int
	contragent_name	varhcar(256)
	contragent_inn	varhcar(12)
Goods (номенклатура товаров)	good_id	int
	good_name	varhcar(50)
Goods (номенклатура товаров)	type_good_id	int
	pict_number	varhcar(50)
	measure_id	int
	good_note	varhcar(50)
	price	float
goods_in_sell (Список проданных товаров)	good_in_sell_id	int
	good_id	int
	journal_sell_id	int
	price_sell	float
	count_goods	int
goods_in_store (товары в наличии)	goods_in_store_id	int
	good_id	int
	count_good	int
	contragent_id	int

Продолжение таблицы 4.1

journal_sell (журнал продаж)	journal_sell_id	int
	date_sell	datetime
journal_sell (журнал продаж)	customer	varchar(256)
	customer_fio	varhcar(50)
	people_id	int
measure (единицы измерения)	measure_id	int
	measure	varhcar(50)
org_attr (сведения об организации)	name_org	varhcar(256)
	org_inn	varchar(12)
	org_kpp	varchar(50)
	adres	varchar(256)
people (сотрудники)	people_id	int
	people_fio	varchar(50)
	dolg	varchar(50)
type_good (вид товара)	type_good_id	int
	type_good	varchar(50)

Исходя из выбранных сущностей и их атрибутов, задается описание типов данных для таблиц базы данных (таблица 4.1).

Диаграмма разрабатываемой БД представлена на рисунке 4.1.

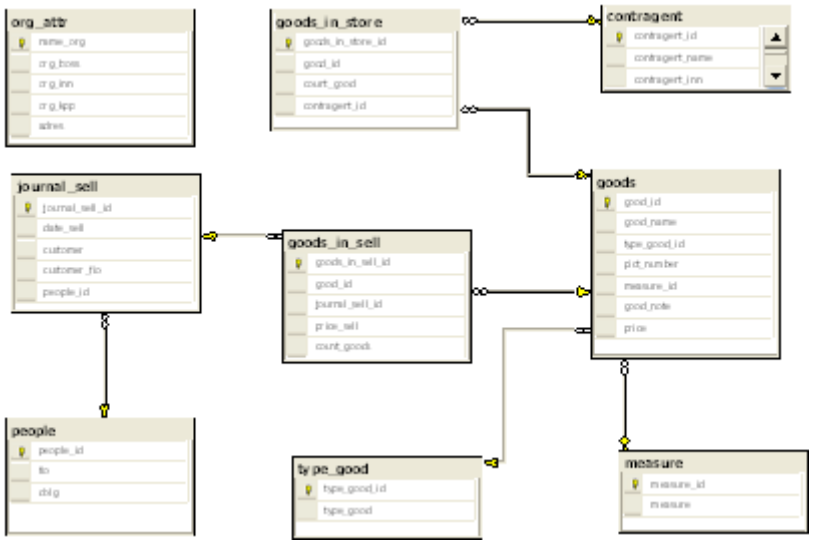

Рисунок 4.1 – Диаграмма разрабатываемой БД

4.2 Структура приложения

Для реализации базы данных отдела продаж будет использована технология клиент-сервер, в которой сервер выполняет только функции хранения данных и защиты данных.

Разрабатываемое приложение выполняет остальные функции.

Подключение к БД осуществляется с использованием класса DBConnection.

Компонент TableAdapter осуществляет наполнение данными из БД на основе введенных запросов или хранимых процедур, помимо этого используется при выполнении операций добавления новых данных, обновления данных в таблицах БД или удаления, а также при сохранении полученных изменений.

С помощью компонента TableAdapter можно выдавать глобальные команды, не связанные ни с одной конкретной таблицей.

Интерфейс пользователя разрабатываемой программы представляет собой приложение, содержащее основное окно, открываемое при запуске приложения, и вызываемые дополнительные.

Для создания окон использовались формы среды разработки Visual Studio 2008.

Главная форма содержит меню, включающее вкладки: Продажи, Товары, Номенклатура, Единицы измерения, Сотрудники, Клиенты, Поставщики, Журнал продаж, а также вкладку поиска.

Спецификация методов класса главной формы представлена в таблице 4.2.

Таблица 4.2 – Спецификация методов класса главной формы

Атрибут	Способ доступа	Назначение
ShowTypeGood	private	Открыть справочник «Вид товара»
ShowPeoples	private	Открыть справочник «Сотрудники»
ShowMeasure	private	Открыть справочник «Единицы измерения»
LoadDictionary	private	Загрузить справочники
ShowGoods	private	Открыть справочник «Товары»
ShowContragent	private	Открыть справочник «Поставщики»
ShowGoodsInStore	private	Показать список товаров на складе
GetMonth	private	Получить месяц в текстовом виде.
GetDayOfWeek	private	Получить день недели в текстовом виде
TreeCalendarBuilding	private	Построить календарное дерево
MainForm_Load	private	Загрузка главной формы. Инициализация объектов связи с базой данных
GetEndCalendarDate	private	Получить конечную дату календарного периода
TreeMouseDown	private	Событие нажатия на элемент календарного дерева
GetStartCalendarDate	private	Получить начальную дату календарного периода
TreeCalendar_AfterSelect	private	Событие выделения элемента календарного дерева
DG_Journal_SelectionChanged	private	Событие выделения строки в журнале продаж
NewSell	private	Новая продажа
ShowSearch	private	Отрыть форму поиска

Управление данными производится посредством выбора вкладки меню в приложении, и далее путем работы в отдельных окнах программы.

Для каждой вкладки создана отдельная форма, что обеспечивает простоту восприятия и использования.

Во время загрузки формы, сохраненные данные из БД заполняют таблицу формы, внесенные изменения при редактировании данных сохранятся в БД при закрытии формы.

Во вкладке поиска данных необходимо задать параметры (слово, часть слова, номер), с помощью компонента типа BindingSource в основном окне программы будет выводиться необходимая информация.

Обобщенный алгоритм работы программы представлен на рисунке 4.2.

Рисунок 4.2 – Обобщенный алгоритм работы программы

Вид главного окна программы показан на рисунке 4.3

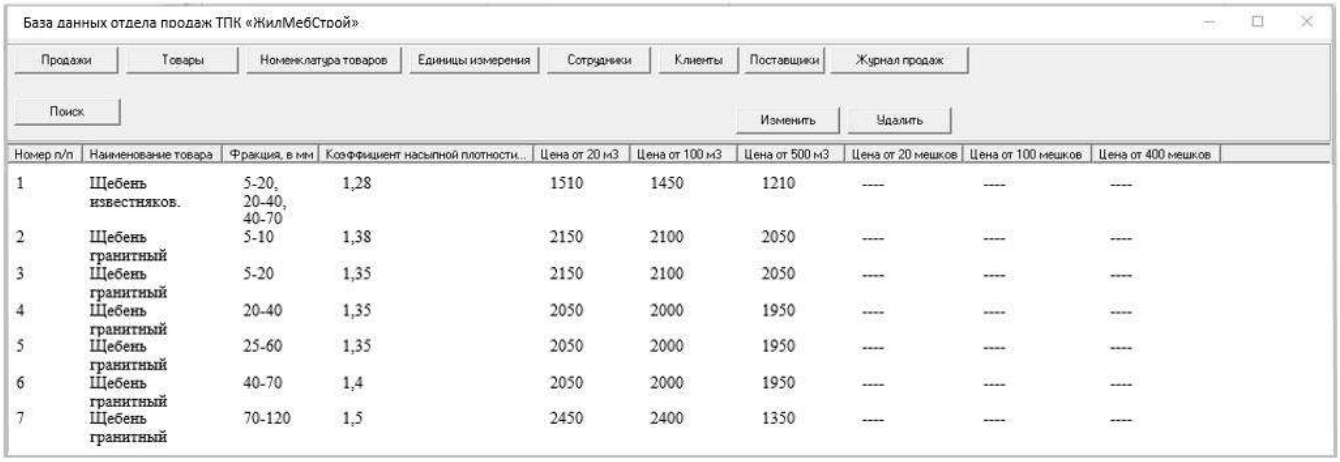

Рисунок 4.3 – Вид главного окна программы

После внесения данных в базу, в области просмотра главного окна можно прочитать информацию прайс-листа. Данные прайс-листа представлены в таблице 4.3.

Таблица 4.3 – Данные прайс-листа

Наименование	Фракция, мм	Коэффициент насыпной плотности, (тн/м3)	Цена от 20 м3	Цена от 100 м3	Цена от 500 м3	Цена от 20 мешков	Цена от 100 мешков	Цена от 400 мешков
Щебень известняков.	5-20, 20-40, 40-70	1,28	1510	1450	1210	----	----	----
Щебень гранитный	5-10	1,38	2150	2100	2050	----	----	----
Щебень гранитный	5-20	1,35	2150	2100	2050	----	----	----
Щебень гранитный	20-40	1,35	2050	2000	1950	----	----	----
Щебень гранитный	25-60	1,35	2050	2000	1950	----	----	----
Щебень гранитный	40-70	1,4	2050	2000	1950	----	----	----
Щебень гранитный	70-120	1,5	2450	2400	1350	----	----	----
Щебень гранитный	80-250	1,5	2450	2400	1350	----	----	----

Продолжение таблицы 4.3

Гранитный отсев	2-5	1,37	2050	2000	1950	----	----	----
Гранитная крошка	0-5	1,37	2550	2500	1450	----	----	----
Щебень гравийный	5-20	1,5	1800	1700	1200	----	----	----
Щебень гравийный	20-40	1,5	1950	1800	1300	----	----	----
Щебень гравийный	40-70	1,5	1950	1800	1300	----	----	----
Песок карьерный	1-1,8	1,5	600	500	400	----	----	----
Песок речной	1,2-1,8	1,6	510	500	480	----	----	----
Песок кварцевый	0,5-0,8	1,6	----	----	----	165	155	150
Песок кварцевый	0,63-1,2 0,8-1,4 0,8-2,0	1,6	----	----	----	170	170	170
Песок кварцевый	2,0-5,0	1,6	----	----	----	190	180	180
Песок сеяный	1,8-2,2	1,65	750	550	455	----	----	----
Песок мытый	2,2-3,0	1,65	495	470	445	----	----	----
Пескогрунт	----	1,5	400	----	----	----	----	----

Алгоритм работы программы для окна поиска представлен на рисунке 4.4.

Рисунок 4.4 – Блок-схема алгоритма работы программы для окна поиска

Вкладка «Продажи» имеет следующие данные:

- Оформить продажу (рисунок 4.5);
- Изменить;
- Удалить запись.

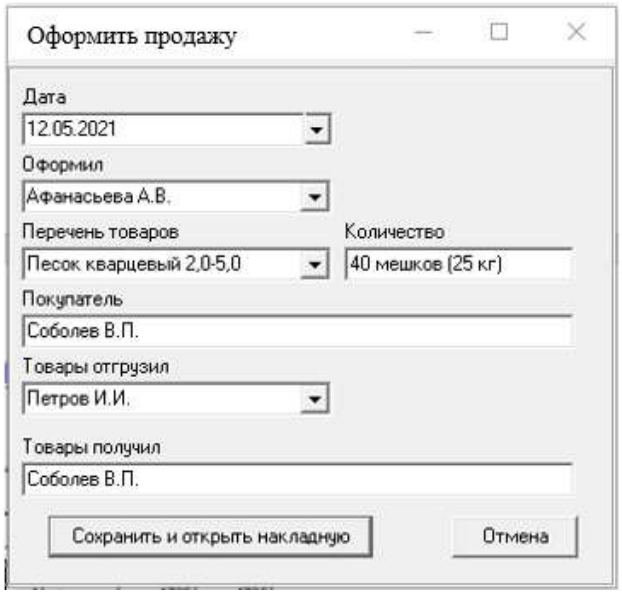

Рисунок 4.5 – Вид окна «Оформить продажу»

Во вкладке «Товары» следующие параметры:

- Идентификационный номер;
- Наименование вида товара;
- Товары в наличии:
  - Номер товара;
  - Количество товаров.

Вид окна «Товары» показан на рисунке 4.6.

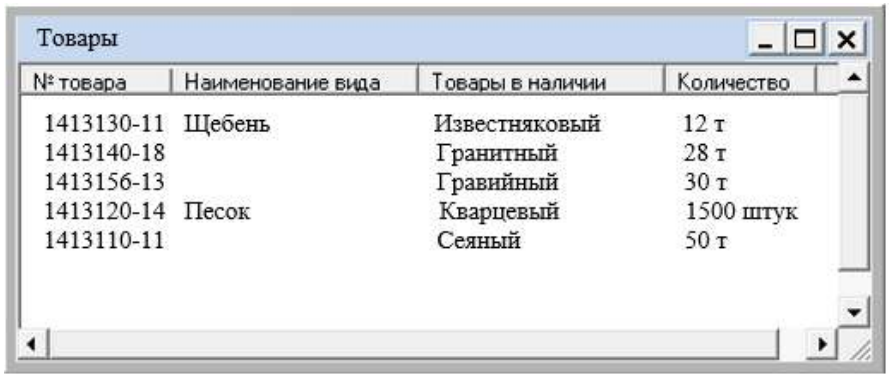

Рисунок 4.6 – Вид окна «Товары»

Вкладка «Единицы измерения» содержит название единиц измерения (м3, т, штуки).

Вкладка «Номенклатура» содержит список номенклатуры товаров в виде:

- 1413000 (Песок, гравий, щебень);
- 1413110-1413124 (Песок природный);
- 1413130-1413172 (Гравий, щебень или дробленый камень);
- Цена.

Вкладка «Сотрудники» содержит список сотрудников отдела продаж, включая следующие данные:

Идентификационный номер;

ФИО,

Должность.

Вкладка «Журнал продаж» содержит данные о продажах:

- Номер записи;
- Дата продажи;
- Наименование организации (покупатель);
- Фамилия, получившего товар;
- Идентификационный номер сотрудника.

Вкладки «Клиенты», «Поставщики» содержат данные об организации:

- Наименование организации;
- ИНН организации.

Отчеты о количестве, объеме продаж, работе сотрудников можно вывести в формате Excell, выбирая по месяцам или сотрудникам.

ЗАКЛЮЧЕНИЕ

В ходе написания работы была разработана база данных информационной подсистемы отдела продаж торгово-промышленной компании. Назначение базы данных – структуризация каталога документов, автоматизация поиска и редактирование данных, формирование контрольных отчетов о количестве, объеме продаж, работе сотрудников.

Разработанная база данных в полной мере удовлетворяет потребности пользователей по составу информации, ее подаче, способах ввода и обработке.

Для разработки приложения был выбран язык программирования C# и среда разработки Visual Studio 2008.

База данных создана с применением программного средства MS SQL Server.

Выбор средства реализации базы данных обусловлен его доступностью, высокой производительностью, простотой работы в нем.

Структура базы данных состоит из 8 таблиц. Основные таблицы полностью включают в себя информацию о работе отдела продаж. Чтобы не перегружать основные таблицы общей информацией, в базе данных сформированы 2 дополнительных таблиц, которые содержат общую и дополняющую информацию. Созданные связи между таблицами позволяют обеспечивать единообразие информации, ее целостность, а также существенно упрощают работу в базе данных. Связи таблиц обеспечены в том числе и посредством выборки данных для заполнения основной таблицы данными из вспомогательной таблицы посредством выпадающего списка. Это предотвращает внесение информации, противоречащей общей концепции базы данных, что может затруднить обработку информации и получения выборочных отчетов, либо иных сводных данных по базе данных.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Биллиг, В. А. Основы программирования на C#; Интернет-университет информационных технологий. Лаборатория знаний – М.: Бином, 2018. – 488 с.

2 Дейт, К. Дж. Введение в системы баз данных: Пер. с англ. 7-е изд. / К. Дж.Дейт − М.; СПб.; Киев: Вильямс, 2017. − 1071 с.

3 Информатика: Базовый курс / под редакцией С. В. Симоновича – СПб.: Питер, 2016. − 258 с.

4 Информатика: Учебник. /Под ред. Н.В. Макаровой. – М.: Финансы и статистика, 2016. − 360 с.

5 Информационные технологии (для экономистов): Учебное пособие / Под общ. ред. А.К. Волкова. – М.: ИНФРА-М, 2017. − 370 с.

6 Карпова, Т.С. Базы данных: модели, разработка, реализация / Т.С. Карпова. – СПб.: Питер, 2016. – 304с.

7 Коротков, Э.М. Концепция менеджмента / Э.М. Коротков. – М.: ДеКА, 2016. – 342с.

8 Кренке, Д. Теория и практика построения баз данных: пер. с англ. / Д. Кренке. – 9-е изд. – СПб.: Питер, 2017. – 858 с.

9 Кузнецов, С. Д. Основы баз данных: учебное пособие // С. Д. Кузнецов – 2-е изд., испр. – М.: Интернет-Университет Информационных Технологий; БИНОМ Лаборатория знаний, 2019. – 484 с., ил.

10 Понамарев, В.В. Программирование на C++/C# в Visual Studio; БХВ-Петербург – М., 2016. – 352 с.

11 Троелсен, Э., C# и платформа .NET. Библиотека программиста / Э. Троэлсон. – СПб.: Питер, 2016. – 796 с.

12 Ульман, Дж. Введение в системы баз данных / Дж. Ульман. – М.: Лори, 2016. – 374с.

13 Хомоненко, А.Д. Базы данных: Учебник для высших учебных заведений / Под ред. проф. А.Д. Хомоненко / А.Д. Хомоненко, В.М. Цыганков, М.Г. Мальцев. – СПб.: КОРОНА принт, 2016. – 416 с.

14 Шилдт, Г. Полный справочник по C# / Г. Шилдт. – М.: Издательский дом «Вильямс», 2017. – 752 с.

15 Обзор СУБД [Электронный ресурс] − URL: https://proglib.io/p/databases-2019/

16 Руководство по MS SQL SERVER [Электронный ресурс] − URL: https://metanit.com/sql/sqlserver/

17 Системные базы данных в MS SQL SERVER [Электронный ресурс] − URL: https://docs.microsoft.com/ru-ru/sql/relational-databases/databases/system-databases?view=sql-server-ver15

ПРИЛОЖЕНИЕ А

Исходный код приложения

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Diagnostics;
using System.Runtime.InteropServices;
using System.Configuration;
using System.Reflection;
using System.IO;

namespace TPK
{
public partial class AddSell : Form
{
DataTable GoodsList = new DataTable();



public AddSell()
{
InitializeComponent();
}

private void AddSell_Load(object sender, EventArgs e)
```

```
{
DB.DoConnection();

measureTableAdapter.Connection = DB.MainConnect;
journal_sellTableAdapter.Connection = DB.MainConnect;
peopleTableAdapter.Connection = DB.MainConnect;
goodsTableAdapter.Connection = DB.MainConnect;
goodStoreTableAdapter.Connection = DB.MainConnect;
goods_in_storeTableAdapter1.Connection = DB.MainConnect;
goods_in_sellTableAdapter1.Connection = DB.MainConnect;
org_attrTableAdapter1.Connection = DB.MainConnect;

this.measureTableAdapter.Fill(this.data_Set.measure); //загрузить данные в таблицу "data_Set.measure"

this.journal_sellTableAdapter.Fill(this.data_Set.journal_sell); // загрузить данные в таблицу "data_Set.journal_sell"
this.peopleTableAdapter.Fill(this.data_Set.people); //загрузить данные в таблицу "data_Set.people"
this.goodsTableAdapter.Fill(this.data_Set.goods); // загрузить данные в таблицу "data_Set.goods"
this.goodStoreTableAdapter.Fill(this.data_Set.GoodStore); //загрузить данные в таблицу "data_Set.GoodStore"

goods_in_storeTableAdapter1.Fill(this.data_Set.goods_in_store);
goods_in_sellTableAdapter1.Fill(this.data_Set.goods_in_sell);
org_attrTableAdapter1.Fill(this.data_Set.org_attr);

GoodsList.Columns.Add("good_id", typeof(int));
```

```
GoodsList.Columns.Add("count_good", typeof(int));
GoodsList.Columns.Add("price_sell", typeof(double));
GoodsList.Columns.Add("price", typeof(double));
GoodsList.Columns.Add("measure_id", typeof(int));

DS_List_good.DataSource = GoodsList;

}

private void CB_Good_SelectedValueChanged(object sender, EventArgs e)
{
if (CB_Good.SelectedValue != null)
{
try
{
COUNT_GOODS.Value = COUNT_GOODS.Maximum = Convert.ToInt32(
data_Set.GoodStore.Compute(" sum(count_good)", "good_id=" +
CB_Good.SelectedValue.ToString()));

PRICE.Text =
data_Set.GoodStore.Compute(" sum(price)", "good_id=" +
CB_Good.SelectedValue.ToString()).ToString();
}
catch { }
}



}
```

```
public string[] SplitText(string Text, int BlockSize) //Разделить строку на блоки
{
if (Text.Length <= BlockSize)
return null;

int SizeStringArray = Text.Length / BlockSize;

if (SizeStringArray * BlockSize < Text.Length)
SizeStringArray++;

string[] StringArray = new string[SizeStringArray];

for (int i = 0; i < SizeStringArray; i++)
{

if ((i * BlockSize + BlockSize) < Text.Length)
StringArray[i] = Text.Substring(i * BlockSize, BlockSize);
else
StringArray[i] = Text.Substring(i * BlockSize, Text.Length - (i * BlockSize));

}

return StringArray;
}

private void AddGoodToList(object sender, EventArgs e)
{
int good_id = Convert.ToInt32(CB_Good.SelectedValue);
```

```
DataRow Row = GoodsList.NewRow();
Row["good_id"] = good_id;
Row["count_good"] = COUNT_GOODS.Value;
Row["price"] = Convert.ToDouble(PRICE.Text);
Row["price_sell"] = Convert.ToDouble( COUNT_GOODS.Value )*
Convert.ToDouble(PRICE.Text);
Row["measure_id"] = 1;

GoodsList.Rows.Add(Row);

if (COUNT_GOODS.Maximum == COUNT_GOODS.Value) //Если выбраны все товары
{
goodStoreBindingSource.Filter += " and (good_id<>" + good_id.ToString()
+ ") ";
}
else
{
DataRow[] Rows = data_Set.GoodStore.Select("good_id=" + good_id.ToString());

foreach (DataRow r in Rows)
{
r["count_good"] = COUNT_GOODS.Maximum - COUNT_GOODS.Value;
}
}
}

public static void PrintReportTableReplaceValues(DataGridView Grid,
```

```
int CellN, //Номер строки с которой выводить таблицу
string TemplateName, //Имя шаблона
DataTable ReplaceValues) //Первая колонка - что заменить, вторая - чем
{
//Инициализация
if (!File.Exists(Application.StartupPath.ToString() + "\\" + TemplateName))
{
// BackPrintExcelTable.ReportProgress(100);
// WaitForm.Close();

MessageBox.Show("При выгрузке данных произошла ошибка. " +
"\nПрограмме не удалось найти файл \n" + Application.StartupPath.ToString()
+ "\\" + TemplateName +
"\nПроверьте наличие указанного файла в директории и попробуйте снова",
"Ошибка открытия файла", MessageBoxButtons.OK, MessageBoxIcon.Error);
return;
}

Application.DoEvents();
// WaitStatusForm progress = new WaitStatusForm("Открытие файла...",
Grid.RowCount + 20, 5);
// progress.Update();

/* try
{ */
string sAppProgID = "Excel.Application";

// Получаем ссылку на интерфейс IDispatch
Type tExcelObj = Type.GetTypeFromProgID(sAppProgID);
```

```
// Запускаем Excel
object oExcel = Activator.CreateInstance(tExcelObj);

object[] args = new object[1];

object oWorkbooks = oExcel.GetType().InvokeMember("Workbooks",
BindingFlags.GetProperty, null, oExcel,
null);

// с принятием всех изменений
//args[0] = true;

args[0] = Application.StartupPath.ToString() + "\\" + TemplateName;

object oWorkbook = oWorkbooks.GetType().InvokeMember("Add",
BindingFlags.InvokeMethod, null, oWorkbooks, args);

object oWorksheets = oWorkbook.GetType().InvokeMember("Worksheets",
BindingFlags.GetProperty, null, oWorkbook, null);
object oWorksheet = oWorksheets.GetType().InvokeMember("Item",
BindingFlags.GetProperty, null, oWorksheets, new object[] { 1 });

// progress.SetValue("Инициализация...", 10);
// progress.Update();
// progress.progressBar1.Value = 10;


int CountExcelColumns = 0; // Сколько будет колонок в экселе
```

```
foreach (DataGridViewColumn col in Grid.Columns)
{
if (col.Visible == true)
CountExcelColumns++;
}

object oCells1_ = oWorksheet.GetType().InvokeMember("Cells",
BindingFlags.GetProperty,
null,
oWorksheet,
new object[] { CellN + 1, 1 });
object oCells2_ = oWorksheet.GetType().InvokeMember("Cells",
BindingFlags.GetProperty,
null,
oWorksheet,
new object[] { CellN + 1, CountExcelColumns });


object oRange_ = oWorksheet.GetType().InvokeMember("Range",
BindingFlags.GetProperty,
null,
oWorksheet,
new object[] { oCells1_, oCells2_ });


oRange_.GetType().InvokeMember("Select", BindingFlags.InvokeMethod,
null, oRange_, null);
object oSelection_ = oExcel.GetType().InvokeMember("Selection",
BindingFlags.GetProperty, null, oExcel,
```

```
null);

object EtireRow = oSelection_.GetType().InvokeMember("EntireRow",
BindingFlags.GetProperty,
null, oSelection_, null);
for (int xxx = 0; xxx < Grid.RowCount; xxx++)
EtireRow.GetType().InvokeMember("Insert", BindingFlags.InvokeMethod,
null, EtireRow, null);
}
```

ПРИЛОЖЕНИЕ Б

Исходный код генерации базы данных

```
USE [master]
GO

/****** Object: Database [TPK] Script Date: 01/05/202117:19:23 ******/
CREATE DATABASE [TPK] ON PRIMARY
(NAME = N'TPK', FILENAME = N'C:\DB\TPK.mdf' , SIZE = 3072KB ,
MAXSIZE = UNLIMITED, FILEGROWTH = 1024KB )
LOG ON
(NAME = N'TPK_log', FILENAME = N'C:\DB\TPK_log.ldf' , SIZE = 10176KB
, MAXSIZE = 2048GB , FILEGROWTH = 10%)
GO

ALTER DATABASE [TPK] SET COMPATIBILITY_LEVEL = 100
GO

IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
begin
EXEC [TPK].[dbo].[sp_fulltext_database] @action = 'enable'
end
GO

ALTER DATABASE [TPK] SET ANSI_NULL_DEFAULT OFF
GO

ALTER DATABASE [TPK] SET ANSI_NULLS OFF
GO

ALTER DATABASE [TPK] SET ANSI_PADDING OFF
```

```
GO

ALTER DATABASE [TPK] SET ANSI_WARNINGS OFF
GO

ALTER DATABASE [TPK] SET ARITHABORT OFF
GO

ALTER DATABASE [TPK] SET AUTO_CLOSE OFF
GO

ALTER DATABASE [TPK] SET AUTO_CREATE_STATISTICS ON
GO

ALTER DATABASE [TPK] SET AUTO_SHRINK OFF
GO

ALTER DATABASE [TPK] SET AUTO_UPDATE_STATISTICS ON
GO

ALTER DATABASE [TPK] SET CURSOR_CLOSE_ON_COMMIT OFF
GO

ALTER DATABASE [TPK] SET CURSOR_DEFAULT GLOBAL
GO

ALTER DATABASE [TPK] SET CONCAT_NULL_YIELDS_NULL OFF
GO
```

```
ALTER DATABASE [TPK] SET NUMERIC_ROUNDABORT OFF
GO

ALTER DATABASE [TPK] SET QUOTED_IDENTIFIER OFF
GO

ALTER DATABASE [TPK] SET RECURSIVE_TRIGGERS OFF
GO

ALTER DATABASE [TPK] SET DISABLE_BROKER
GO

ALTER DATABASE [TPK] SET AUTO_UPDATE_STATISTICS_ASYNC
OFF
GO

ALTER DATABASE [TPK] SET DATE_CORRELATION_OPTIMIZATION
OFF
GO

ALTER DATABASE [TPK] SET TRUSTWORTHY OFF
GO

ALTER DATABASE [TPK] SET ALLOW_SNAPSHOT_ISOLATION OFF
GO

ALTER DATABASE [TPK] SET PARAMETERIZATION SIMPLE
GO
```

```
ALTER DATABASE [TPK] SET READ_COMMITTED_SNAPSHOT OFF
GO

ALTER DATABASE [TPK] SET HONOR_BROKER_PRIORITY OFF
GO

ALTER DATABASE [TPK] SET READ_WRITE
GO

ALTER DATABASE [TPK] SET RECOVERY SIMPLE
GO

ALTER DATABASE [TPK] SET MULTI_USER
GO

ALTER DATABASE [TPK] SET PAGE_VERIFY CHECKSUM
GO

ALTER DATABASE [TPK] SET DB_CHAINING OFF
GO


USE [TPK]
GO
/****** Object: Table [dbo].[type_good] Script Date: 01/05/2021 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
```

```
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[type_good](
	[type_good_id] [int] IDENTITY(1,1) NOT NULL,
	[type_good] [varchar](50) NULL,
CONSTRAINT [PK_type_good] PRIMARY KEY CLUSTERED
(
	[type_good_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/****** Object: Table [dbo].[people] Script Date: 01/05/2021 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[people](
	[people_id] [int] IDENTITY(1,1) NOT NULL,
	[fio] [varchar](50) NULL,
	[dolg] [varchar](50) NULL,
CONSTRAINT [PK_people] PRIMARY KEY CLUSTERED
(
```

```
	[people_id] ASC
	)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
	) ON [PRIMARY]
	GO
	SET ANSI_PADDING OFF
	GO
	/****** Object: Table [dbo].[org_attr] Script Date: 01/05/2021 ******/
	SET ANSI_NULLS ON
	GO
	SET QUOTED_IDENTIFIER ON
	GO
	SET ANSI_PADDING ON
	GO
	CREATE TABLE [dbo].[org_attr](
		[name_org] [varchar](256) NULL,
		[org_boss] [varchar](256) NULL,
		[org_inn] [varchar](256) NULL,
		[org_kpp] [varchar](50) NULL,
		[adres] [varchar](256) NULL
	) ON [PRIMARY]
	GO
	SET ANSI_PADDING OFF
	GO
	/****** Object: Table [dbo].[measure] Script Date: 01/05/2021 ******/
	SET ANSI_NULLS ON
	GO
	SET QUOTED_IDENTIFIER ON
```

```
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[measure](
	[measure_id] [int] IDENTITY(1,1) NOT NULL,
	[measure] [varchar](50) NULL,
CONSTRAINT [PK_measure] PRIMARY KEY CLUSTERED
(
	[measure_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/****** Object: Table [dbo].[contragent] Script Date: 01/05/2021 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[contragent](
	[contragent_id] [int] IDENTITY(1,1) NOT NULL,
	[contragent_name] [varchar](256) NULL,
	[contragent_inn] [varchar](25) NULL,
CONSTRAINT [PK_contragent] PRIMARY KEY CLUSTERED
(
```

```
	[contragent_id] ASC
	)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
	) ON [PRIMARY]
	GO
	SET ANSI_PADDING OFF
	GO
	/****** Object: Table [dbo].[goods] Script Date: 01/05/2021 ******/
	SET ANSI_NULLS ON
	GO
	SET QUOTED_IDENTIFIER ON
	GO
	SET ANSI_PADDING ON
	GO
	CREATE TABLE [dbo].[goods](
		[good_id] [int] IDENTITY(1,1) NOT NULL,
		[good_name] [varchar](50) NULL,
		[type_good_id] [int] NULL,
		[pict_number] [varchar](50) NULL,
		[measure_id] [int] NULL,
		[good_note] [varchar](50) NULL,
		[price] [float] NULL,
	CONSTRAINT [PK_goods] PRIMARY KEY CLUSTERED
	(
		[good_id] ASC
	)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
```

```
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/****** Object: Table [dbo].[journal_sell] Script Date: 01/05/2021 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[journal_sell](
	[journal_sell_id] [int] IDENTITY(1,1) NOT NULL,
	[date_sell] [datetime] NULL,
	[customer] [varchar](256) NULL,
	[customer_fio] [varchar](256) NULL,
	[people_id] [int] NULL,
CONSTRAINT [PK_journal_sell] PRIMARY KEY CLUSTERED
(
	[journal_sell_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/****** Object: Table [dbo].[goods_in_store] Script Date: 01/05/2021 ******/
SET ANSI_NULLS ON
```

```
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[goods_in_store](
	[goods_in_store_id] [int] IDENTITY(1,1) NOT NULL,
	[good_id] [int] NULL,
	[count_good] [int] NULL,
	[contragent_id] [int] NULL,
CONSTRAINT [PK_goods_in_store] PRIMARY KEY CLUSTERED
(
	[goods_in_store_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
/****** Object: Table [dbo].[goods_in_sell] Script Date: 01/05/2021 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[goods_in_sell](
	[goods_in_sell_id] [int] IDENTITY(1,1) NOT NULL,
	[good_id] [int] NULL,
	[journal_sell_id] [int] NULL,
	[price_sell] [float] NULL,
	[count_goods] [int] NULL,
CONSTRAINT [PK_goods_in_sell] PRIMARY KEY CLUSTERED
(
```

```
[goods_in_sell_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
/****** Object: Default [DF_goods_in_store_count_good] Script Date:
01/05/2021 ******/
ALTER TABLE [dbo].[goods_in_store] ADD CONSTRAINT
[DF_goods_in_store_count_good] DEFAULT ((0)) FOR [count_good]
GO
/****** Object: ForeignKey [FK_goods_measure] Script Date: 01/05/2021
******/
ALTER TABLE [dbo].[goods] WITH CHECK ADD CONSTRAINT
[FK_goods_measure] FOREIGN KEY([measure_id])
REFERENCES [dbo].[measure] ([measure_id])
GO
ALTER TABLE [dbo].[goods] CHECK CONSTRAINT [FK_goods_measure]
GO
/****** Object: ForeignKey [FK_goods_type_good] Script Date:
01/05/2021******/
ALTER TABLE [dbo].[goods] WITH CHECK ADD CONSTRAINT
[FK_goods_type_good] FOREIGN KEY([type_good_id])
REFERENCES [dbo].[type_good] ([type_good_id])
GO
ALTER TABLE [dbo].[goods] CHECK CONSTRAINT [FK_goods_type_good]
GO
/****** Object: ForeignKey [FK_goods_in_sell_goods] Script Date:
01/05/2021******/
```

```
ALTER TABLE [dbo].[goods_in_sell] WITH CHECK ADD CONSTRAINT
[FK_goods_in_sell_goods] FOREIGN KEY([good_id])
REFERENCES [dbo].[goods] ([good_id])
GO
ALTER TABLE [dbo].[goods_in_sell] CHECK CONSTRAINT
[FK_goods_in_sell_goods]
GO
/****** Object: ForeignKey [FK_goods_in_sell_journal_sell] Script Date:
01/05/2021******/
ALTER TABLE [dbo].[goods_in_sell] WITH CHECK ADD CONSTRAINT
[FK_goods_in_sell_journal_sell] FOREIGN KEY([journal_sell_id])
REFERENCES [dbo].[journal_sell] ([journal_sell_id])
GO
ALTER TABLE [dbo].[goods_in_sell] CHECK CONSTRAINT
[FK_goods_in_sell_journal_sell]
GO
ALTER TABLE [dbo].[goods_in_store] WITH CHECK ADD CONSTRAINT
[FK_goods_in_store_contragent] FOREIGN KEY([contragent_id])
REFERENCES [dbo].[contragent] ([contragent_id])
GO
ALTER TABLE [dbo].[goods_in_store] CHECK CONSTRAINT
[FK_goods_in_store_contragent]
GO
/****** Object: ForeignKey [FK_goods_in_store_goods] Script Date:
01/05/2021******/
ALTER TABLE [dbo].[goods_in_store] WITH CHECK ADD CONSTRAINT
[FK_goods_in_store_goods] FOREIGN KEY([good_id])
REFERENCES [dbo].[goods] ([good_id])
GO
```

```
ALTER TABLE [dbo].[goods_in_store] CHECK CONSTRAINT
[FK_goods_in_store_goods]
GO
ALTER TABLE [dbo].[journal_sell] WITH CHECK ADD CONSTRAINT
[FK_journal_sell_people] FOREIGN KEY([people_id])
REFERENCES [dbo].[people] ([people_id])
GO
ALTER TABLE [dbo].[journal_sell] CHECK CONSTRAINT
[FK_journal_sell_people]
GO
```