**СОДЕРЖАНИЕ**

[ТЕРМИНЫ, ОПРЕДЕЛЕНИЯ, ИСПОЛЬЗУЕМЫЕ СОКРАЩЕНИЯ 1](#_Toc96639648)

[ВВЕДЕНИЕ 3](#_Toc1299639554)

[1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 4](#_Toc2063475420)

[1.1 Описание предметной области 5](#_Toc1042955780)

[1.2 Функции и организационная структура информационной системы 7](#_Toc101333667)

[1.3 Описание потоков данных и бизнес-процессов 12](#_Toc128480953)

[1.4 Обзор и анализ проектных решений 19](#_Toc48234285)

[2. СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ 25](#_Toc1660555659)

[2.1 Разработка концепции и архитектуры построения и платформы реализации информационной системы 25](#_Toc2022005525)

[2.2 Техническое обеспечение ИС 29](#_Toc2005902448)

[3. ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ СИСТЕМЫ 36](#_Toc2040518577)

[3.1 Концептуальное проектирование базы данных 36](#_Toc881842207)

[3.2 Выделение ключей. Переход от концептуальной модели к реляционной логической модели данный 37](#_Toc501549619)

[3.3 Нормализация отношений и поддержка целостности данных 39](#_Toc466789135)

[3.4 Построение физической модели 52](#_Toc2085534975)

[3.5Проектирование SQL-запросов 57](#_Toc894548654)

# ТЕРМИНЫ, ОПРЕДЕЛЕНИЯ, ИСПОЛЬЗУЕМЫЕ СОКРАЩЕНИЯ

|  |  |
| --- | --- |
| Обозначение | Описание |
| ИС | Информационная система кафе |
| БД | База данных |
| ОС | Операционная система |
| ЛВС | Локальная вычислительная сеть |
| СКС | Структурированная кабельная система |
| СУБД | Система управления базами данных |
| ГОСТ | Государственный стандарт |
| ФЗ | Федеральный закон |
| *IDEF0* | Графическая нотация, предназначенная для формализации и описания бизнес-процессов |
| *DFD* | Диаграмма потоков данных |
| *UML* | Унифицированный язык моделирования |
| *SQL* | Декларативный язык программирования |

**ВВЕДЕНИЕ**

Тема курсового проекта: "Оффлайн магазин автозвука". Выбор данной темы обусловлен необходимостью разработки системы, которая обеспечит эффективное управление магазином автозвука. Владельцам магазинов требуется инструмент для учета товаров, ведения продаж и обеспечения контроля над работой персонала с целью повышения уровня обслуживания клиентов.

Актуальность данной темы подчеркнута не только стремлением оптимизировать работу розничных точек, но и улучшением качества обслуживания покупателей. Система учета позволит улучшить оперативность предоставления информации о наличии товаров, сократить время обслуживания клиентов, а также повысить контроль за эффективностью работы персонала.

Целью проведения исследования в рамках курсового проекта является разработка и реализация системы для оффлайн магазина автозвука. Разрабатываемая система должна обеспечивать учет товаров, автоматизацию процесса продажи и контроль за эффективностью работы персонала.

В контексте современных вызовов в розничной торговле, эта система может способствовать повышению эффективности работы магазинов и улучшению обслуживания клиентов.

При разработке данного проекта использовалась База Данных PostgreSQL, а также интегрированная программная среда pgAdmin для разработки таблиц базы данных, SQL-запросов, триггеров и хранимых процедур.

Выбор PostgreSQL обоснован его широким функционалом, открытым исходным кодом, а также высокой степенью надежности и производительности. PostgreSQL обеспечивает эффективное взаимодействие с серверным приложением, а поддержка стандарта SQL делает его удобным инструментом для работы с базой данных.

Использование PostgreSQL обеспечивает легкий доступ к серверному приложению через запросы на языке SQL и другие средства, предоставляемые этой базой данных. Это позволяет эффективно реализовать функциональность системы, обеспечивая удобство работы с данными и их адекватное хранение в базе данных.

1. **АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ**

## 1.1 Описание предметной области

Оффлайн магазин автозвука представляет собой торговое учреждение, специализирующееся на предоставлении товаров для автомобильного звукового оборудования. В данной организации управление базой данных становится ключевым элементом для эффективной организации работы и предоставления высококачественного обслуживания клиентов.

Деловой регламент:

1. После обращения клиента в магазин автозвука его личные данные (ФИО, контактный номер телефона) вводятся в таблицу "Клиенты". Это позволяет вести учет покупок, предпочтений, и совершать оперативные продажи, а также обеспечивает возможность осуществления связи для предоставления консультаций;
2. Информация о продавцах, работающих в области автозвука и данных о производителе хранится в таблице "Производители". Это важно для обеспечения клиентов квалифицированным обслуживанием и оптимизации рабочего процесса;
3. Товары, предоставляемые магазином, фиксируются в таблице "Товары". Здесь указываются наименования товаров, их наличие в магазине, и стоимость, что облегчает учет имеющегося ассортимента и управление ценообразованием;
4. Записи о покупках товаров регистрируются в таблице "Покупки". Они содержат информацию о клиенте, купленном товаре, дате и времени покупки, а также сумме сделки;
5. Информация об оплате покупки клиентом фиксируется в таблице "Оплата". Это включает сумму оплаты и указание номера покупки для проведения транзакции, что обеспечивает контроль над финансовым процессом;
6. Информация о товарах, которые клиент приобретает, а также их количестве и стоимости, сохраняется в таблице "Товары в покупке". Это позволяет вести точный учет предпочтений каждого клиента, оптимизировать ассортимент товаров в соответствии с популярностью;
7. Также клиент может сделать заказ товара со склада, если в магазине его недостаточно, запись о заказе хранится в таблице «Заказы», когда клиент оплатит заказ, статус этого заказа изменится, а данные об оплате будут храниться в таблице «Оплата заказа».

Основные бизнес-процессы, которые можно выделить в рамках предложенной предметной области:

ОБРАЩЕНИЕ КЛИЕНТА В МАГАЗИН. При посещении магазина автозвука новым клиентом, его контактные данные (ФИО, контактный номер телефона) регистрируются в базе данных. Каждый клиент получает уникальный идентификатор для более эффективного учета и предоставления персонализированных предложений.

ВЫБОР ТОВАРА. При выборе товара, клиент определяется с конкретным товаром, количество которого ему необходимо приобрести. Далее, он уточняет стоимость товара в соответствии с ценами, указанными в таблице "Товары".

ОФОРМЛЕНИЕ ПОКУПКИ. После консультации и выбора товара клиентом, оформляется запись в таблице "Покупки", где фиксируются данные о клиенте, купленном товаре, дате и времени покупки, а также стоимости сделки.

ОПЛАТА ПОКУПКИ. Когда клиент завершает покупку, он осуществляет оплату в соответствии с указанной стоимостью товара. Данные об оплате фиксируются в таблице "Оплата", включая сумму, и номер покупки для точного проведения транзакции.

ОФОРМЛЕНИЕ ЗАКАЗА. Если в магазине товара не достаточно для совершения покупки, клиент может заказать этот товар со склада.

ОПЛАТА ЗАКАЗА. Данные об оплате заказа хранятся в таблице «Оплата заказа», так же когда поступила оплата, статус заказа автоматически изменяется.

В процессе выполнения курсовой работы необходимо реализовать выполнение следующих требований:

1. Регистрация личных данных нового клиента в базе данных магазина автозвука, включая ФИО, контактный номер телефона, с присвоением уникального идентификатора для каждого клиента;
2. Предоставление клиенту возможности выбора из ассортимента автозвука и внесение данных о выбранном товаре, количестве, стоимости в таблицу "Товары в покупке";
3. Оформление покупки в магазине автозвука после консультации с клиентом, внесение данных о клиенте, купленном товаре, дате и времени покупки, а также стоимости сделки в таблицу "Покупки";
4. Оплата покупки клиентом в соответствии с указанной стоимостью товара. Фиксация данных об оплате, включая сумму и номер покупки, в таблице "Оплата" для последующего учета транзакции. Если стоимость товара изменилась, оплата за товар автоматически корректируется;
5. Оформление заказа при нехватке наличия определенного товара в магазине.
6. Оплата заказа клиентом в соответствии со стоимостью заказа, после поступления оплаты, в таблице «Заказы» статус заказа изменяется.

1.2 [Функции и организационная структура информационной системы](https://docs.google.com/document/d/11lQKgR3OPMJSS0-VLQc0xtAmJX4d4Mci/edit" \l "heading=h.2et92p0)

Далее рассмотрим организационную структуру магазина автомобильной акустики. Организационную структура представляет структуру системы - иерархию организационных подразделений, должностей и конкретных лиц, многообразие связей между ними, а также территориальную привязку структурных подразделений. Организационное представление является взглядом на тех, кто участвует в выполнении бизнес-процессов предприятия, и на те ресурсы, которые при этом задействуются. Оптимальное структурирование организации зависит от множества факторов, поэтому универсальной организационной структуры-прототипа не существует. Целью анализа организационной структуры является выявление обоснованного количества уровней иерархии; выявление подчинения одних и тех же звеньев (сотрудников) различным руководителям, наличия малого количества подчиненных у одного руководителя и другое. Схема организационной структуры магазина представлена на рисунке 1.1. У магазина автомобильной акустики есть отделы: бухгалтерский, маркетинга, взаимодействия с клиентами, ремонта и технического обслуживания.

За работу магазина автомобильной акустики отвечает генеральный директор. У него в подчинение находятся отделы: бухгалтерский и кадровый. Отдельно в подчинении находятся управляющий магазина автомобильной акустики.

Основные задачи бухгатерского отдела:

* ведение учета затрат на закупки;
* ведение учета зарплатного фонда;
* расчет цен на продукцию;
* уточнение цен на продукцию при изменении ситуации на рынке.

Основные задачи кадрового отдела:

* поиск персонала;
* ведение информации о сотрудниках;
* планирование карьерного плана для сотрудников.

Основные задачи отдела работы с клиентами:

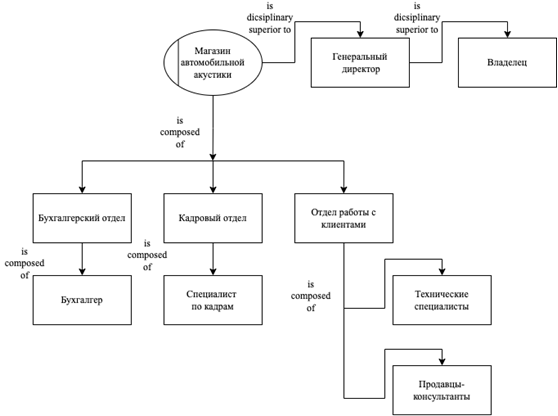
* организация и контроль работы с клиентами, реализации товаров осуществления осмотров и выписывания направлений на процедуры;
* консультирование клиентов;
* осуществление медицинских осмотров.

Функциональная структура предметной области представлена на рисунке 1.1.



Рисунок 1.1. – Функциональная структура предметной области.

Далее рассмотрим организационную структуру систему магазина автомобильной акустики(рис 1.2) с точки зрения матричной модели (Таблица 1.1). Матричная модель использует комбинацию функционального деления и деления по группам клиентов, что позволяет более обстоятельно решать вопросы управления.

 Рисунок 1.2 – Организационная структура

Два сотрудника (руководителя), отвечающих за свой участок деятельности, имеют возможность, высказать свою точку зрения и прийти к соглашению, в большей степени отвечающему потребностям клиента. Использование двухмерной матричной модели, комбинирующей функциональное деление, обычно приводит, с одной стороны, к формированию подразделений, осуществляющих те или иные операции, и служб, осуществляющих продажу информационного контента.

Таблица 1.1 – Матричная модель организационной структуры

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Работы | Исполнители | | | | | |
|  | Бухгалтерский отдел | | | Кадровый отдел | | Отдел работы с клиентами |
| ведение учета затрат на закупки | + | | |  | |  |
| ведение учета зарплатного фонда | + | | |  | |  |
| расчет цен на продукцию | + | | |  | |  |
| уточнение цен на продукцию при изменении ситуации на рынке | + | | |  | |  |
| поиск персонала |  | | | + | |  |
| ведение информации о сотрудниках |  | | | + | |  |
| планирование карьерного плана для сотрудников |  | | | + | |  |
| обеспечение маркетинговых исследований |  | | | + | |  |
| организация и контроль работы с клиентами, реализации товаров осуществления осмотров и выписывания направлений на процедуры | |  |  | | + | |
| консультирование клиентов | |  |  | | + | |
| осуществление медицинских осмотров | |  |  | | + | |

## 1.3 Описание потоков данных и бизнес-процессов

Для построения контекстной диаграммы определим главный бизнес-процесс Системы и его роли: роли управления, роли входа, роли выхода и роли механизма. Главным бизнес-процессом Системы является “Работа магазина автомобильной акустики”.

Входными данными является обращение клиента. На выходе бизнес процесса - отчет о реализации, ведомость по номенклатуре на складах, чек об оплате, прибыль и проданный товар. В качестве управления бизнес-процесса выступают: инструкция пользователя управления приложением; закон о персональных данных 152-ФЗ с 31.12.2017, список товаров и устав фирмы. В роли механизмов будут выступать: персонал и техническое оборудование. Диаграмма показывающая описание данной бизнес функции изображена на рисунке 1.3.

Произведем разбиение сложного процесса на составляющие его функции. Для этого декомпозируем процесс А0 на процессы А1, А2, А3. Эти разбиения продемонстрированы диаграммой декомпозиции главного бизнес-процесса. Диаграмма представлена на рисунке 1.4.

Процесс А1 – Взаимодействие с поставщиком. На входе процесс получает обращение клиента. Работа выполняется под управлением инструкции пользователя управления приложения, устава магазина автомобильной акустики и списка товаров, а также с применением «механизмов» персонала и технического оборудования. В результате работы получают выходные ресурсы - ведомость по номенклатуре на складах и поступивший товар.

Процесс А2 – Взаимодействие с клиентом. На вход процесс получает поступивший товар. Работа выполняется под управлением инструкции пользователя управления приложения и устава магазина автомобильной акустики, Закона о защите персональных данных 152-ФЗ от 31.12.2017, с применением «механизмов» персонала и технического оборудования. В результаты работы получаются выходной ресурс: заявка клиента.

Процесс А3 – Продажа товара. На вход процесс получает заявку клиента. Работа выполняется под управлением инструкции пользователю управления приложением и устава магазина автомобильной акустики и списка товаров с применением «механизмов» персонала и технического оборудования. В результате работы на выходе процесс получает: Чек об оплате, Проданный товар, Отчет о реализации, Прибыль.

Процесс А1 можно декомпозировать на три процесса. Процесс А1.1 – Контакт с поставщиком, процесс А1.2 – Оформление документа о заказе, А1.3 – Принятие на склад. Диаграмма представлена на рисунке 1.5.

Процесс А2 можно декомпозировать на два процесса. Процесс А2.1 – Консультация клиента, А2.2 – Ввод данных клиента, А2.3 - Оформление заявки. Декомпозиция процесса представлена на рисунке 1.6.

Процесс А3 можно декомпозировать на два процесса. Процесс А3.1 – Оформление документа о продаже, А3.2 – Оплата товара, А3.3 – Передача товара. Декомпозиция процесса представлена на рисунке 1.7.

Далее рассмотрим структуру магазина автомобильной акустики на схеме в стандарте *DFD*. Источники информации (внешние сущности) порождают информационные потоки (потоки данных), переносящие информацию к подсистемам или процессам, как это сказано в [3],[4]. Те в свою очередь преобразуют информацию и порождают новые потоки, которые переносят информацию к другим процессам или подсистемам, накопителям данных или внешним сущностям - потребителям информации. На диаграмме, представленной на рисунке 1.8, смоделированы потоки данных главного бизнес-процесса магазина автомобильной акустики.

* внешние сущности – показывают входы и выходы их системы. Именем является существительное. Могут выступать физические лица, материальные предметы, которые находятся за границами ИС;
* работы – показывают операции, проводимые над входными данными, которые в дальнейшем преобразуются в какой-либо материальный или нематериальный объект;
* накопитель данных представляет собой абстрактное устройство для хранения информации, которую можно в любой момент поместить в накопитель и через некоторое время извлечь, причем способы помещения и извлечения могут быть любыми.
* поток данных определяет информацию, передаваемую через некоторое соединение от источника к приемнику. Реальный поток данных может быть информацией, передаваемой по кабелю между двумя устройствами, пересылаемыми по почте письмами, магнитными лентами или дискетами, переносимыми с одного компьютера на другой и т.д., обозначается стрелкой.

Основными элементами проектируемой системы являются следующие компоненты:

* внешние сущности – “Клиент”, “Управляющий”, “Кладовщик”,

“Контрагент”, “Консультант”;

* работы – “Контакт с поставщиком”, “Доставка товаров на склад”,

“Консультация клиента”, “Оформление продажи”, “Оформление документа о продаже”, “Оплата товара”, “Выдача товара”;

* хранилища данных - “Номенклатура”, “Контрагенты”,

“Сотрудники”, “Документы о поступлении”, “Склад”.

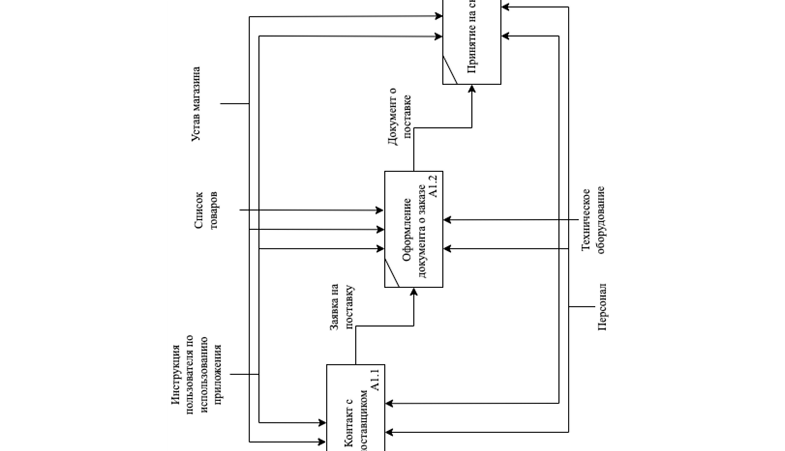
Изображение выглядит как текст, диаграмма, снимок экрана, линия

Автоматически созданное описание

Рисунок 1.3 – Главная бизнес-функция ИС

Изображение выглядит как диаграмма, текст, План, Технический чертеж

Автоматически созданное описание Рисунок 1.4 – Декомпозиция главной бизнес-функции ИС

 Рисунок 1.5 – Декомпозиция подфункции “Взаимодействие с поставщиком”

Изображение выглядит как текст, диаграмма, Технический чертеж, План

Автоматически созданное описание

Рисунок 1.6 – Декомпозиция подфункции “Взаимодействие с клиентом

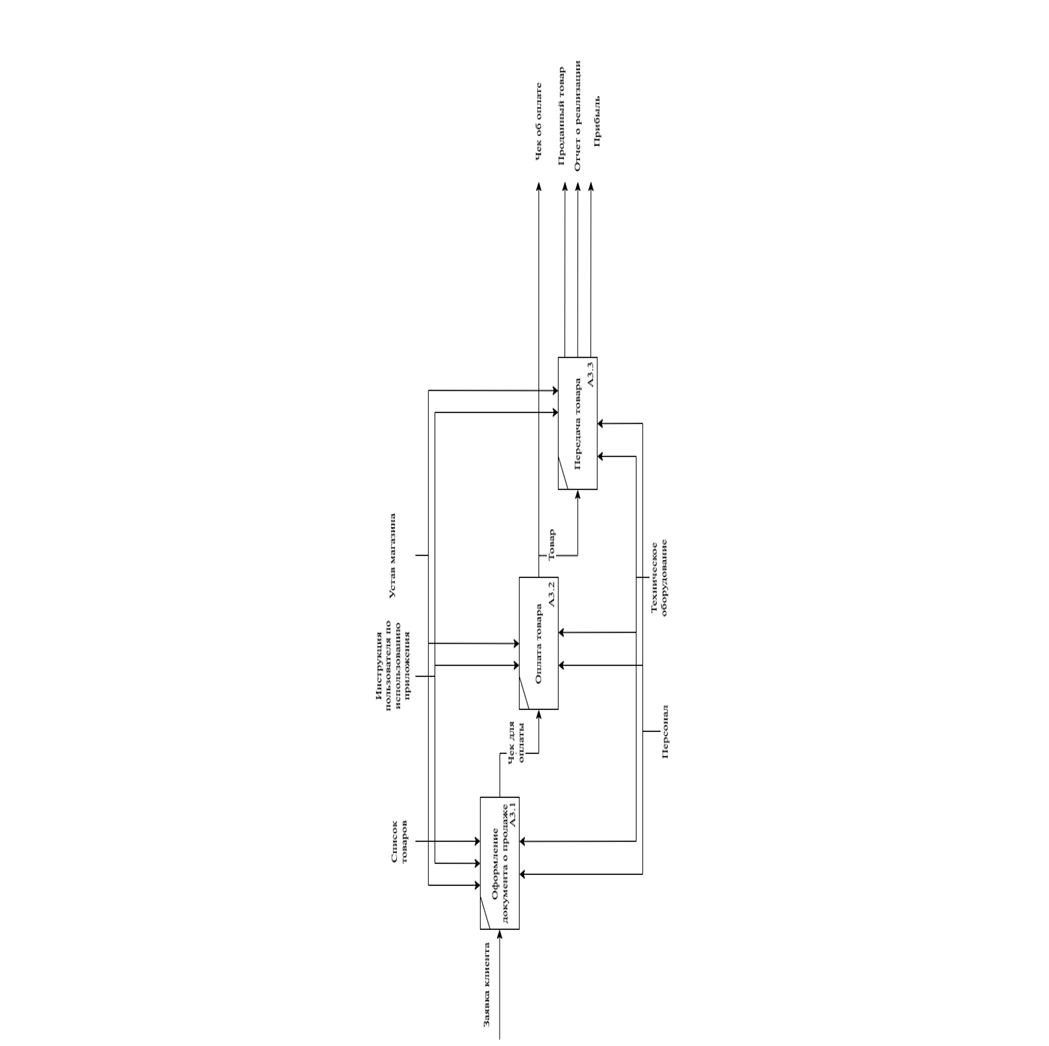


Рисунок 1.7 – Декомпозиция подфункции “Продажа товара”

Изображение выглядит как текст, диаграмма, План, Технический чертеж

Автоматически созданное описание

Рисунок 1.8 – Диаграмма потоков данных процесса “Взаимодействие с поставщиком”

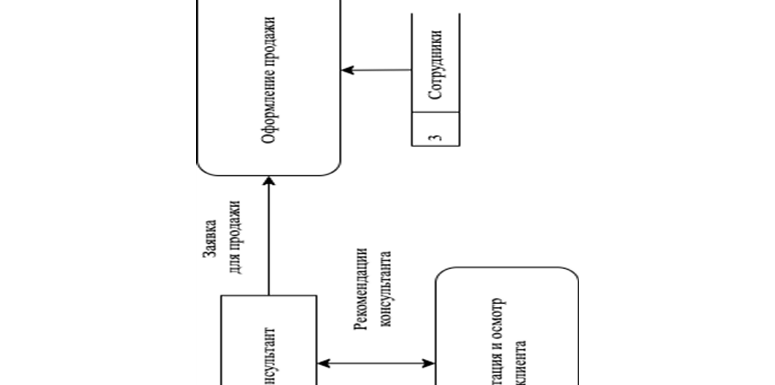


Рисунок 1.9 – Диаграмма потоков данных процесса “Взаимодействие с клиентом”

Изображение выглядит как диаграмма, План, текст, Технический чертеж

Автоматически созданное описаниеРисунок 1.10 – Диаграмма потоков данных процесса “Продажа товара”

## 1.4 Обзор и анализ проектных решений

Наиболее популярными системами, используемыми в настоящее время для автоматизации торговой и складской деятельности предприятий, а также и в анализируемом магазине автомобильной акустики, являются программы "1С: Торговля и склад" и "1С: Управление торговлей".

"1С: Управление торговлей 8" - это современный инструмент повышения эффективности бизнеса торгового предприятия. Прикладное решение позволяет в комплексе автоматизировать задачи оперативного и управленческого учета, анализа и планирования торговых операций, обеспечивая тем самым эффективное управление современным торговым предприятием.

"1C: Торговля и склад" предназначена для учета любых видов торговых операций. Благодаря гибкости и настраиваемости система способна выполнять все функции учета - от ведения справочников и ввода первичных документов до получения различных ведомостей и аналитических отчетов.

"1C: Торговля и склад" автоматизирует работу на всех этапах деятельности предприятия и позволяет:

* вести раздельный управленческий и финансовый учет;
* вести учет от имени нескольких юридических лиц;
* вести партионный учет товарного запаса с возможностью выбора метода списания себестоимости (FIFO, LIFO, по средней);
* вести раздельный учет собственных товаров и товаров, взятых на реализацию;
* оформлять закупку и продажу товаров;
* производить автоматическое начальное заполнение документов на основе ранее введенных данных;
* вести учет взаиморасчетов с покупателями и поставщиками, детализировать взаиморасчеты по отдельным договорам;
* выполнять резервирование товаров и контроль оплаты;
* вести учет денежных средств на расчетных счетах и в кассе;
* вести учет товарных кредитов и контроль их погашения;
* вести учет переданных на реализацию товаров, их возврат и оплату.

При использовании программы "1C: Торговля и склад" пользователь может:

* задать для каждого товара необходимое количество цен разного типа, хранить цены поставщиков, автоматически контролировать и оперативно изменять уровень цен;
* работать с взаимосвязанными документами;
* выполнять автоматический расчет цен списания товаров;
* быстро вносить изменения с помощью групповых обработок справочников и документов;
* вести учет товаров в различных единицах измерения;
* получать самую разнообразную отчетную и аналитическую информацию о движении товаров и денег;
* автоматически формировать бухгалтерские проводки для 1C: Бухгалтерии.

"1C: Торговля и склад" содержит средства обеспечения сохранности и непротиворечивости информации:

* возможность запрещения пользователям прямого удаления информации;
* специальный режим удаления данных с контролем перекрестных ссылок;
* возможность запрещения пользователям редактировать данные за прошлые отчетные периоды;
* установка запрета на редактирование печатных форм документов;
* запирание системы пользователем при временном прекращении работы.

Система "1C: Торговля и склад" наряду со своими плюсами имеет ряд недостатков:

* требует больших денежных затрат на приобретение;
* установить систему "1C: Торговля и склад" может не любой пользователь;
* для работы с системой персонал должен пройти специализированные курсы.

Рассмотренные программы являются основными и широко используемые программы для торговли и складского хозяйства. Так же существует большое разнообразие похожих программ. Например:

Программа "БЭСТ-4 МАГАЗИН" автоматизирует работу торгового зала с различными электронными кассовыми аппаратами и POS-терминалами. Чеки выбиваются на основании электронных прайс-листов, в которых отражены все необходимые характеристики товара. Эта программа отличается развитыми функциями учета движения товаров на складе и в торговом зале, обеспечивает работу со счетами-фактурами и автоматическое формирование книг закупок и продаж ТМЦ, партионный учет и продажу товаров комплектами.

Программный комплекс "Гепард" (Windows) предназначена для комплексной автоматизации различных видов торговой деятельности. Существует два варианта поставки: для розничной и оптовой торговли.

В состав комплекса в настоящее время входят модули:

* Администратор комплекса;
* Товарный склад;
* Отдел продаж;
* Прайс-листы;
* Анализ товарного склада;
* Книга продаж;
* Книга закупок;
* Себестоимость контрактов;
* Общая бухгалтерия;
* Финансовый анализ;
* Касса;
* Банк;
* Платежные поручения;
* Основные средства;
* Материалы;
* Регистрация счетов-фактур;
* Делопроизводство.

Программа "ФолиоWin" ориентирована на различные группы пользователей:

* ФолиоWin-Склад 4 - для малых предприятий;
* ФолиоWin 7 - для крупных и средних предприятий (на платформе MS-SQL Server);
* ФолиоWin7Е - малых и средних предприятий (MSDE- версия) Две последние являются клиент-серверными продуктами, которые обладают следующими возможностями:
* ввод данных с 5-50 рабочих мест при интенсивной работе (основан на особенностях архитектуры клиент-сервер);
* многоуровневая система разграничения доступа, полная конфиденциальность информации, высокая надежность;
* встроенный генератор отчетов и форм, дополнительная библиотека отчетов и документов;
* одновременная работа со многими складами как с одним складом, среди прочих включающая функции: ввода в один документ товаров из разных складов, установки общих скидок и цен, перерасчета учетных цен, автоподбора партий к оплате, получения сводной много складской отчетности;
* дополнительные возможности по оптимизации и быстродействию комплекса с помощью программного модуля ФОЛИО-Мультисервер (поставляется отдельно).

Программа "Оборот+ версии 7.0" предназначена для ведения складского и финансового учета в магазине, оптовой фирме, на складе. Она автоматизирует все складские, финансовые операции.

"SLS- Склад" - эта программа при ведении учета основных средств позволяет на каждое основное средство оформить карточку учета с подробным описанием характеристик и отражать историю постановки на учет, начисления амортизации, переоценки и выбытия.

"Домино 8" - от компании Софт-Вест разработана для автоматизации торговых организаций. Она сочетает в себе простоту и удобство работы, гибкость настройки и глубину анализа деятельности предприятия, позволяет контролировать и эффективно использовать все имеющиеся ресурсы.

Она помогает поддерживать оптимальный ассортимент товарных запасов, формировать заказы на поставку товаров, получать всю необходимую бухгалтерскую отчетность. На ее основе имеется несколько отраслевых решений:

* ДОМИНО: Магазин. Для предприятий розничной торговли как одиночных, так и объединенных в сеть;
* ДОМИНО: Торговый Дом. Для торговых холдингов;
* ДОМИНО: Фаст-Фуд- Для предприятий быстрого питания, баров, кафе;
* ДОМИНО: Дисконтный клуб. Для управления лояльностью покупателей;
* ДОМИНО: Центр досуга. Для кинотеатров, концертных залов, центров досуга.

1. **СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ**

## 2.1 Разработка концепции и архитектуры построения и платформы реализации информационной системы

Согласно предпроектному анализу предметной области, одной из ключевых целей разработки является снижение затрат пользователей, планирующих приобретать товары из разрабатывающего онлайн платформы. Разрабатываемый проект информационной системы предназначен для мобильных устройств на платформе Android версии 7.0 и выше. Согласно [9,10], Android – это платформа для мобильных технологий, обеспечивающая мобильные устройства мощностью и мобильностью операционной системы Linux, надежностью и мобильностью стандартного языка высокого уровня и интерфейса прикладного программирования, а также обширным набором полезных приложений. В настоящее время считается, что платформа Android занимает почти три четверти мирового рынка смартфонов.

На рисунке 2.1 представлена схема функционирования и построения информационной системы. При построении данной схемы учтены особенности архитектуры разрабатываемой системы. Разрабатываемое приложение обладает трехуровневой архитектурой, компонентами которой являются:

- слой представления;

- слой приложения;

- слой доступа к данным.

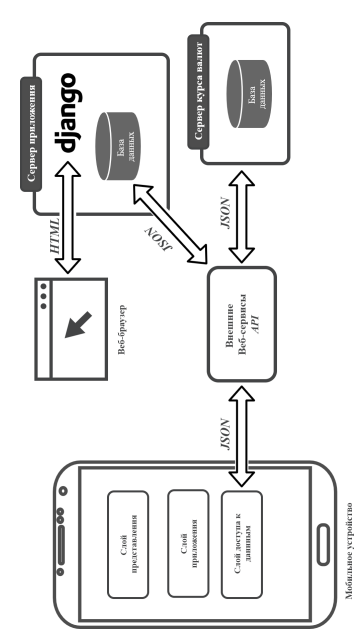


Рисунок 2.1 – Схема функционирования и построения информационной системы

К простейшей бизнес-логике можно отнести: интерфейс авторизации, алгоритмы шифрования, проверка корректности вводимых пользователем данных и элементарные операции. На слое приложения сосредоточена реализация всей бизнес-логики приложения, кроме элементов передаваемых клиенту и элементов, реализующих подключение к базе данных. Данный слой должен обеспечивать горизонтальное масштабирование разрабатываемого приложения.

Слой доступа к данным обеспечивает подключение к серверу информационной базы (ИБ). Для установления подключения задаются параметры сервера, добавляются соответствующие разрешения. Для физической реализации сервера ИБ выбран Django rest framework (DRF), работа с которым подробно описана на официальном сайте [11]. Django — это веб-среда, написанная на языке высокого уровня Python, которая способствует быстрой разработке и обладающая открытым исходным кодом. Для администрирования ИБ, реализованной на DRF необходим только Веб-браузер, установка дополнительного ПО не требуется. Несмотря на наличие своей собственной номенклатуры, такой как присвоение имен вызываемым объектам, генерирующим HTTP-ответы, «views», базовая структура Django может рассматриваться как архитектура MVC. Более подробное определение понятий «framework» и «MVC» приведено в книге «The Django Book. Release 2.0» [12]. Состав MVC включает объектно-реляционный преобразователь (ORM), который является посредником между моделями данных (определяемыми как классы Python) и реляционной базой данных («Model»), системой для обработки HTTP-запросов с помощью системы веб-шаблонов («View») и диспетчер URL («Controller»). Взаимодействие серверов ИБ и курса валют с приложением, установленным на устройстве пользователя, осуществляется с помощью формата обмена данными JavaScript Object Notation (JSON), который использует текст для хранения и передачи объектов данных, состоящих из пар 32 атрибут-значение. JSON — это независимый от языка программирования формат данных, созданный на основе языка программирования JavaScript.

Синтаксис формата JSON подробно описан в книге «ECMA-404: The JSON Data Interchange Format» [13]. Для получения информации о текущем курсе валют используется сервер конвертера валют free.currencyconverterapi [14]. Данный сервер бесплатно предоставляет информацию о текущем курсе валют в формате JSON. Подключение к серверу обеспечивается с помощью уникального ключа, предоставляемого разработчику. Ответы в формате JSON, полученные от серверов передаются на слой доступа данным через интерфейс прикладного программирования (API) - вычислительный интерфейс, определяющий взаимодействие между приложением и сервером. API определяет виды вызовов или запросов, которые могут быть сделаны, как их выполнять, форматы данных, которые следует использовать, соглашения, которым необходимо следовать.

## 2.2 Техническое обеспечение ИС

В комплекс технических средств должны входить следующие элементы: – сервер БД; – ПК пользователей; – ПК администраторов. Требования к техническим характеристикам серверов БД:

– процессор

– Intel Xeon E-2236 3.4ГГц

– объем оперативной памяти – 16 Гб;

– дисковая подсистема

– 4 х 512 Гб;

– устройство чтения компакт-дисков (DVD-ROM);

– сетевой адаптер – 1000 Мбит.

Требования к техническим характеристикам ПК пользователя и ПК администратора: – процессор – AMD Ryzen 5 3600; – объем оперативной памяти – 8 Гб; – дисковая подсистема – 500 Гб; – устройство чтения компакт-дисков (DVD-ROM); – сетевой адаптер – 1000 Мбит. План размещения вычислительной техники представлен на рисунке 2.3. Расположение рабочих помещений на рисунке 2.3: 1 - вестибюль; 2 - комната охраны; 3, 10, 11 – санузел; 4, 5, 6 – зал мероприятий; 7 – коридор; 8, 9, 14, 15, 16 – кабинет; 12 – серверная; 13 – кухня.

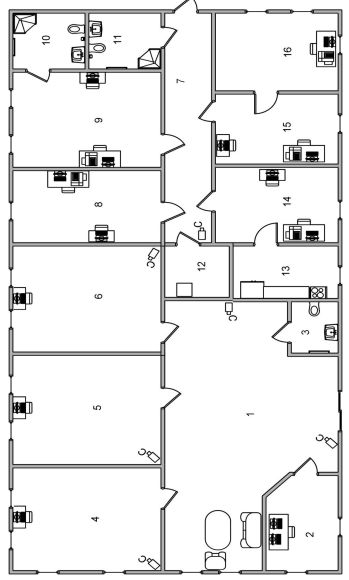


Рисунок 2.3 – План размещения вычислительной техники

В ходе работы над ВКР была рассмотрена топология "звезда" для структурированной кабельной системы, как оптимальная по стоимости, так и наиболее удобная с точки зрения последующего администрирования, проанализировав материалы [6], [7].

Таким образом, ЛВС должна иметь топологию «звезда» с ядром в серверном помещении. В ЛВС детского развивающего клуба будет использоваться стандарт Gigabit Ethernet (1000 Мбит/сек). При построении ЛВС на характер разводки влияют различные критерии: доступная стоимость проекта, технические характеристики, объемы и направления информационных потоков, расположение рабочих помещений.

В числе названных, последняя в наибольшей степени определяет топологию. Небольшая сеть с компактным расположением помещений в пределах одного этажа может быть разведена через единственный коммутатор вблизи сервера. Детский развивающий клуб имеет один этаж с небольшой площадью помещения, поэтому для организации сети был выбран вариант разведения сети через один коммутатор 3 уровня.

В качестве сервера для детского развивающего клуба был выбран HP ProLiant DL180 Gen9, выбор обоснован тем, что данный сервер характеризуется высокой надежностью и долговечностью, способен справляться с высокими нагрузками, а так же в дальнейшем при расширении компании есть возможность задействовать свободные отсеки для дисков системы хранения.

В качестве активного оборудования был выбран коммутатор HUAWEI S5720-28X-LI-AC, который имеет 24 Ethernet-порта 10/100/1000 Base-T, 4 порта 10Gig SFP и маршрутизатор HUAWEI AR1220EVW, который имеет 8 портов GE (с возможностью настройки в качестве WAN-интерфейсов). На рисунке 2.4 представлена схема локальной вычислительной сети детского развивающего клуба.

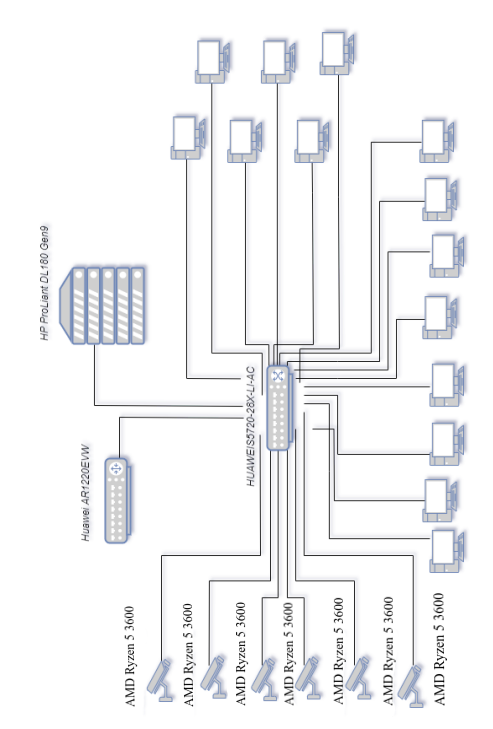


Рисунок 2.4 – Схема локальной вычислительной сети

Все кабели ЛВС разводятся на новые патч-панели, устанавливаемые в коммутационном шкафу. Количество гнезд на патч-панелях определяется количеством портов ЛВС, учитывая возможность расширения не менее чем на 20%. Количество и длина необходимых патч-кордов (RJ-45/RJ-45) для коммутации информационных розеток определяется исходя из количества гнезд на патч-панелях.

В коммутационных шкафах необходимо придерживаться следующего расположения. Сверху вниз: вентилятор, патч-панели, активное оборудование, сервер, источники бесперебойного питания [8].

Все медные компоненты СКС должны соответствовать стандартам EIA/TIA-568-B и ISO/IEC 11801 для медных кабельных систем класса D (категория 5е) и обеспечивать передачу данных на скоростях не менее 1 Гбит/сек.

Наше здание состоит из одного этажа, на котором необходимо проложить кабель для подключения рабочих станций к сети. Наиболее подходящий тип кабеля - неэкранированная витая пара категории 5e.

Таким образом, был выбран кабель Siemon 9C5L4-E2. На рисунке 2.5 представлена структурированная кабельная система детского развивающего клуба.

Было рассчитано необходимое количество кабеля для построения сети детского развивающего клуба.

Для вычисления количества кабеля существует два метода: метод суммирования и эмпирический метод.

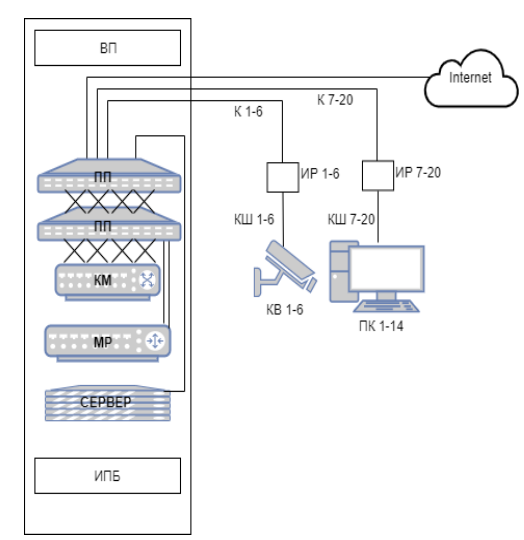


Рисунок 2.5 – Структурированная кабельная система детского досугового центра

1) Метод суммирования

Метод суммирования заключается в подсчете длины трассы каждого горизонтального кабеля с последующим сложением этих длин. К полученному результату добавляется технологический запас величиной до 10%, а также запас для выполнения разделки в розетках и на кроссовых панелях. В таблице 2.3 представлен расчет длины кабеля методом суммирования.

Общая длина кабеля с учетом технологического запаса в 10 % составила: 189 + 189 0.1 = 207,9м

2) Эмпирический метод Сущность эмпирического метода заключается в применении для подсчета длины кабеля обобщенной эмпирической формулы:



где – средняя длина кабельных трасс;

и – длина самого близкого и далекого участка;

– коэффициент технологического запаса (1,1м) X – запас для разделки кабеля (0,3м)

Количество кабельных пробросов, на которые хватает катушки кабеля:

где Lcb – длина кабельной катушки (305м)

Рассчитав необходимое количество кабеля двумя методами, было принято решение на создание СКС использовать большее из значений, а именно 305 метра. Это значение было получено путем вычисления методом суммирования.

1. **ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ СИСТЕМЫ**

## 3.1 Концептуальное проектирование базы данных

Концептуальная модель базы данных представлена на рисунке 3.1.

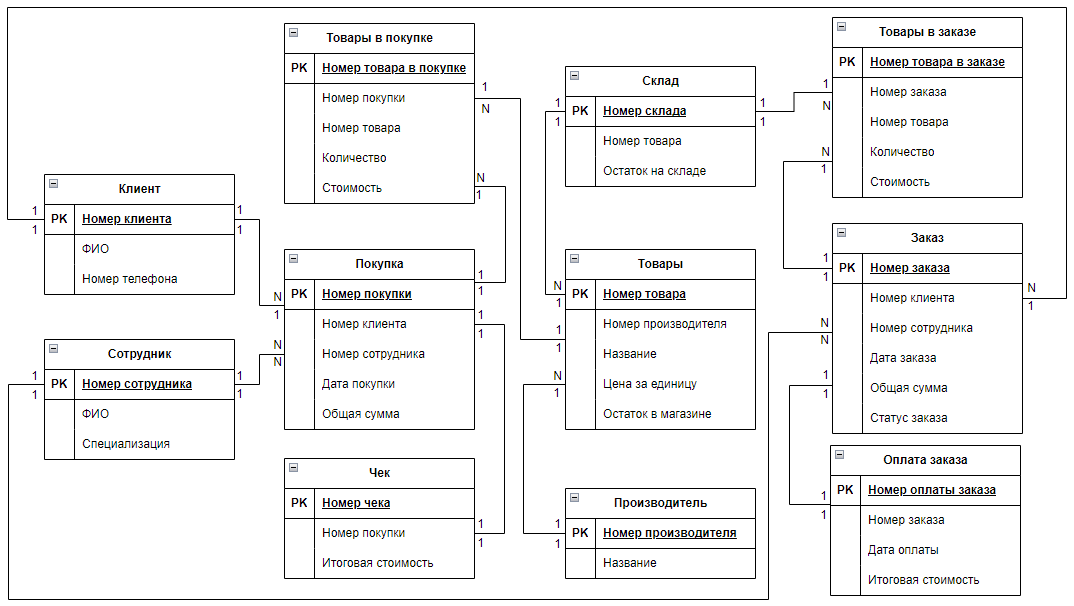


Рисунок 3.1. – Концептуальная модель данных.

Анализируя полученную концептуальную модель базы данных, можно прийти к выводу о том, что полученная база данных является достаточно простой и очевидной, что нивелирует необходимость использования специальных средств оценки структурных характеристик БД. Так же, исходя из концептуальной модели можно сделать вывод, что отсутствует необходимость введения дополнительных таблиц либо же нормализации БД каким-либо другим способом. В модели отсутствуют связи N:N. Сохранены только связи 1:n и 1:1, что видно из рисунка.

## 3.2 Выделение ключей. Переход от концептуальной модели к реляционной логической модели данный

На рисунке 3.2 представлена логическая модель данных.

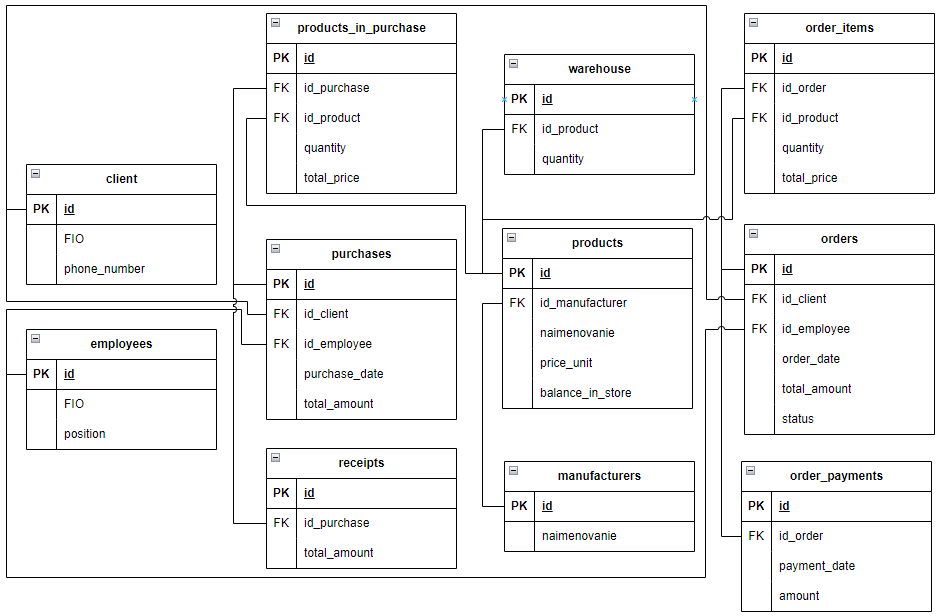


Рисунок 3.2 – Логическая модель данных.

Исходя из построенной модели, можно выделить первичные, внешние и родительские ключи для каждой из таблиц.

Таблица «Client»: поле «id» - первичный ключ.

Таблица «Emloyees»: поле «id» - первичный ключ.

Таблица «Manufacturers»: поле «id» - первичный ключ.

Таблица «Products»: поле «id» - первичный ключ, а поле «id\_manufacturer» - внешний ключ, для которого поле «id» таблицы «Manufacturers» является родительским.

Таблица «Purchases»: поле «id» - первичный ключ. Поле «id\_client» - внешний ключ, для которого поле «id» таблицы «Client» является родительским. Поле «id\_employee» - внешний ключ, для которого поле «id» таблицы «Employees» является родительским.

Таблица «Product\_in\_purchase»: поле «id» - первичный ключ, а поле «id\_purchase» - внешний ключ, для которого поле «id» таблицы «Purchases» является родительским. Поле «id\_product» - внешний ключ, для которого поле «id» таблицы «Products» является родительским.

Таблица «Receipts»: поле «id» - первичный ключ, а поле «id\_purchase» - внешний ключ, для которого поле «id» таблицы «Purchases» является родительским.

Таблица «Warehouse»: поле «id» - первичный ключ, а поле «id\_product» - внешний ключ, для которого поле «id» таблицы «Products» является родительским.

Таблица «Orders»: поле «id» - первичный ключ, а поле «id\_client» - внешний ключ, для которого поле «id» таблицы «Clients» является родительским. Поле «id\_employee» - внешний ключ, для которого поле «id» таблицы «Employees» является родительским.

Таблица «Order\_items»: поле «id» - первичный ключ, а поле «id\_order» - внешний ключ, для которого поле «id» таблицы «Orders» является родительским. Поле «id\_product» - внешний ключ, для которого поле «id» таблицы «Products» является родительским.

Таблица «Order\_payments»: поле «id» - первичный ключ, а поле «id\_order» - внешний ключ, для которого поле «id» таблицы «Orders» является родительским.

Так как структура модели достаточно проста, при проектировании концептуальной модели не возникло необходимости в изменении структуры таблиц. Данная модель изначально соответствовала всем 6 правилам перехода от концептуальной модели к реляционной логической модели данных.

## 3.3 Нормализация отношений и поддержка целостности данных

В процессе нормализации отношений необходимо привести все отношения в нормальную форму.

Нормальная форма — свойство отношения в реляционной модели данных, характеризующее его с точки зрения избыточности, потенциально приводящей к логически ошибочным результатам выборки или изменения данных. Нормальная форма определяется как совокупность требований, которым должно удовлетворять отношение.

Нормализация предназначена для приведения структуры БД к виду, обеспечивающему минимальную логическую избыточность, и не имеет целью уменьшение или увеличение производительности работы или же уменьшение, или увеличение физического объёма базы данных. Конечной целью нормализации является уменьшение потенциальной противоречивости, хранимой в базе данных информации.

Общее назначение процесса нормализации заключается в следующем:

1. исключение некоторых типов избыточности;
2. устранение некоторых аномалий обновления;
3. разработка проекта базы данных, который является достаточно «качественным» представлением реального мира, интуитивно понятен и может служить хорошей основой для последующего расширения;
4. упрощение процедуры применения необходимых ограничений целостности.

Устранение избыточности производится, как правило, за счёт декомпозиции отношений таким образом, чтобы в каждом отношении хранились только первичные факты.

В связи с небольшим объемом таблиц и грамотным подходом к конструированию моделей данных, проекту не требуется нормализация отношений. В проекте отсутствуют аномалии вставки, удаления и обновления данных в таблицах.

Главная особенность SQL-технологий наличие у сервера СУБД специальных средств контроля целостности данных, не зависящих от клиентских программ и привязанных непосредственно к таблицам. Т.е. принципиально не важно, каким образом осуществляется доступ к базе данных: через SQL-консоль, через ODBC-драйвера из приложения Windows, через WWW-connector из Internet-браузера или через DBI-интерфейс Perl. В любом из этих случаев, за контролем целостности данных следит сервер, и при нарушении правил целостности данных сервер известит клиента об ошибке.

К структурам контроля целостности данных относятся ограничители (constraint), которые привязаны к столбцам и триггеры (trigger), которые могут быть привязаны как к столбцам, так и к строкам в таблице.

Ограничители - это элементарные проверки или условия, которые выполняются для операций вставки и модификации значения столбца. Если данная проверка не проходит или условие не выполняется, то вставка или модификация отменяется, а в программу клиента передается ошибка.

SQL-серверы, как правило, поддерживают следующие ограничители:

* NOT NULL - проверка на непустое значение.
* NULL - специальное понятие в СУБД, которое означает "пусто".
* UNIQUE - проверка на уникальность. Вставляемое значение должно быть уникально для данного столбца по всей таблице. Может содержать пустые значения.
* SERIAL - используется для создания столбца, который автоматически увеличивается с каждой новой вставкой записи. Это часто используется для создания уникальных идентификаторов (первичных ключей) в таблицах.
* PRIMARY KEY - первичный ключ. Значение в столбце считается первичным ключом, если оно непустое и уникально в пределах столбца данной таблицы. Первичный ключ может быть составным и представлять собой комбинацию столбцов. Тогда чтобы считаться первичным ключом, каждое из группы значений не должно быть пустыми и формируемые строки значений первичного ключа должны быть уникальны в пределах таблицы. Первичный ключ - основа для построения индексов по таблице.

SQL-технология позволяет на уровне столбца задавать домены значений, т.е. строго определенные наборы или диапазоны значений, для помещаемых в столбец данных. В частности, можно реализовывать ограничения ссылочной целостности (referential integrity constraint) и проверки фиксированного условия. Ограничение ссылочной целостности не позволяет значениям из столбца одной таблицы принимать значения кроме как из присутствующих в столбце другой таблицы. Это делается при помощи ограничителей FOREIGN KEY (внешний ключ) и REFERENCES (указатель ссылки). Таблица, содержащая FOREIGN KEY, считается родительской таблицей. Таблица, содержащая REFERENCES, считается дочерней таблицей. Внешний ключ и указатель ссылки могут находиться в одной таблице, т.е. родительская таблица одновременно является дочерней.

* FOREIGN KEY - внешний ключ. Назначает столбец или комбинацию столбцов в текущей (родительской) таблице в качестве внешнего ключа для ссылки из других таблиц.
* REFERENCES - указатель ссылки (или родительский ключ). Указывает на столбец (комбинацию столбцов) в родительской таблице, ограничивающую значения в текущей (дочерней) таблице.

Необходимо, до добавления новой строки в требуемую таблицу, сгенерировать первичный ключ для данного кортежа с целью обеспечения целостности объектов. Первичные ключи генерируют последовательности, которые создаются с помощью ключевого слова SERIAL после уникального идентификатора, при создании таблицы.

Текст триггеров:

Триггер для автоматического обновления остатка в магазине в таблице «Products» при добавлении товара в покупку:

*CREATE OR REPLACE FUNCTION update\_product\_balance\_on\_purchase()*

*RETURNS TRIGGER AS $$*

*BEGIN*

*-- Обновление остатка в магазине*

*UPDATE products*

*SET balance\_in\_store = balance\_in\_store - NEW.quantity*

*WHERE id = NEW.id\_product;*

*RETURN NEW;*

*END;*

*$$ LANGUAGE plpgsql;*

*CREATE TRIGGER trigger\_update\_product\_balance\_on\_purchase*

*BEFORE INSERT ON products\_in\_purchase*

*FOR EACH ROW*

*EXECUTE FUNCTION update\_product\_balance\_on\_purchase();*

Триггер для автоматического вычисления итоговой стоимости по количеству товаров:

*CREATE OR REPLACE FUNCTION update\_total\_price()*

*RETURNS TRIGGER AS $$*

*BEGIN*

*SELECT price\_unit \* NEW.quantity INTO NEW.total\_price*

*FROM products*

*WHERE id = NEW.id\_product;*

*RETURN NEW;*

*END;*

*$$ LANGUAGE plpgsql;*

*CREATE TRIGGER calculate\_total\_price*

*BEFORE INSERT OR UPDATE OF quantity ON products\_in\_purchase*

*FOR EACH ROW*

*EXECUTE FUNCTION update\_total\_price();*

Триггер который автоматически подставляет цену в таблицу «Покупка», суммируя все «total\_price» в таблице «Товары в покупке», для определенной покупки:

*CREATE OR REPLACE FUNCTION update\_total\_amount()*

*RETURNS TRIGGER AS $$*

*BEGIN*

*UPDATE purchases*

*SET total\_amount = (SELECT COALESCE(SUM(total\_price), 0) FROM products\_in\_purchase WHERE id\_purchase = NEW.id\_purchase)*

*WHERE id = NEW.id\_purchase;*

*RETURN NEW;*

*END;*

*$$ LANGUAGE plpgsql;*

*CREATE TRIGGER calculate\_total\_amount*

*AFTER INSERT OR UPDATE OF total\_price ON products\_in\_purchase*

*FOR EACH ROW*

*EXECUTE FUNCTION update\_total\_amount();*

Триггер который автоматически создает чек при создании покупки и подставляет сумму исходя из общей суммы в таблице "Покупка":

*CREATE OR REPLACE FUNCTION update\_receipt\_total\_amount()*

*RETURNS TRIGGER AS $$*

*BEGIN*

*-- Если вставляются новые данные в таблицу purchases*

*IF TG\_OP = 'INSERT' THEN*

*INSERT INTO receipts (id\_purchase, total\_amount)*

*VALUES (NEW.id, NEW.total\_amount);*

*-- Если изменяется сумма в таблице purchases*

*ELSIF TG\_OP = 'UPDATE' AND NEW.total\_amount <> OLD.total\_amount THEN*

*UPDATE receipts*

*SET total\_amount = NEW.total\_amount*

*WHERE id\_purchase = NEW.id;*

*END IF;*

*RETURN NEW;*

*END;*

*$$ LANGUAGE plpgsql;*

*CREATE TRIGGER generate\_receipt\_for\_purchase*

*AFTER INSERT OR UPDATE OF total\_amount ON purchases*

*FOR EACH ROW*

*EXECUTE FUNCTION update\_receipt\_total\_amount();*

Триггер который обеспечивает автоматическое обновление стоимости в таблицах «Товары в покупке», «Покупка», «Чек», «Товары в заказе», «Заказ», «Оплата заказа» - при изменении стоимости товара в таблице «Товары»:

*CREATE OR REPLACE FUNCTION update\_purchase\_and\_receipt\_amount()*

*RETURNS TRIGGER AS $$*

*BEGIN*

*-- Если изменяется цена товара в таблице products*

*IF TG\_OP = 'UPDATE' AND NEW.price\_unit <> OLD.price\_unit THEN*

*-- Обновляем сумму в таблице products\_in\_purchase*

*UPDATE products\_in\_purchase*

*SET total\_price = NEW.price\_unit \* quantity*

*WHERE id\_product = NEW.id;*

*-- Обновляем сумму в таблице order\_items*

*UPDATE order\_items*

*SET total\_price = NEW.price\_unit \* quantity*

*WHERE id\_product = NEW.id;*

*-- Обновляем общую сумму в таблице purchases*

*UPDATE purchases*

*SET total\_amount = COALESCE((SELECT SUM(total\_price) FROM products\_in\_purchase WHERE id\_purchase = purchases.id), 0)*

*WHERE id = ANY(ARRAY(SELECT id\_purchase FROM products\_in\_purchase WHERE id\_product = NEW.id));*

*-- Обновляем общую сумму в таблице orders*

*UPDATE orders*

*SET total\_amount = COALESCE((SELECT SUM(total\_price) FROM order\_items WHERE id\_order = orders.id), 0)*

*WHERE id = ANY(ARRAY(SELECT id\_order FROM order\_items WHERE id\_product = NEW.id));*

*-- Обновляем общую сумму в таблице order\_payments*

*UPDATE order\_payments*

*SET amount = orders.total\_amount*

*FROM orders*

*WHERE order\_payments.id\_order = orders.id;*

*-- Обновляем сумму в таблице receipts*

*UPDATE receipts*

*SET total\_amount = purchases.total\_amount*

*FROM purchases*

*WHERE receipts.id\_purchase = purchases.id;*

*END IF;*

*RETURN NEW;*

*END;*

*$$ LANGUAGE plpgsql;*

*CREATE TRIGGER update\_purchase\_and\_receipt\_amount\_trigger*

*AFTER UPDATE OF price\_unit ON products*

*FOR EACH ROW*

*EXECUTE FUNCTION update\_purchase\_and\_receipt\_amount();*

Триггер на проверку наличия товара при покупке:

*CREATE OR REPLACE FUNCTION check\_product\_availability()*

*RETURNS TRIGGER AS $$*

*BEGIN*

*-- Проверяем наличие товара в таблице products*

*IF NEW.quantity > (SELECT balance\_in\_store FROM products WHERE id = NEW.id\_product) THEN*

*RAISE EXCEPTION 'Недостаточно товара в магазине';*

*END IF;*

*RETURN NEW;*

*END;*

*$$ LANGUAGE plpgsql;*

*CREATE TRIGGER check\_product\_availability\_trigger*

*BEFORE INSERT ON products\_in\_purchase*

*FOR EACH ROW*

*EXECUTE FUNCTION check\_product\_availability();*

Триггер который отвечает за то, чтобы при изменении данных в таблице «Клиент», информация о клиенте в таблице «Покупка» и «Заказ» изменялась автоматически:

*CREATE OR REPLACE FUNCTION update\_purchase\_and\_order\_client()*

*RETURNS TRIGGER AS $$*

*BEGIN*

*-- Обновляем данные в таблице purchases при изменении данных в таблице client*

*UPDATE purchases*

*SET id\_client = NEW.id*

*WHERE id\_client = OLD.id;*

*-- Обновляем данные в таблице orders при изменении данных в таблице client*

*UPDATE orders*

*SET id\_client = NEW.id*

*WHERE id\_client = OLD.id;*

*RETURN NEW;*

*END;*

*$$ LANGUAGE plpgsql;*

*CREATE TRIGGER update\_purchase\_and\_order\_client\_trigger*

*AFTER UPDATE ON client*

*FOR EACH ROW*

*EXECUTE FUNCTION update\_purchase\_and\_order\_client();*

Триггер который отвечает за то, чтобы при изменении данных в таблице «Сотрудники», информация о сотруднике в таблице «Покупка» и «Заказ» изменялась автоматически:

*CREATE OR REPLACE FUNCTION update\_purchase\_and\_order\_employee\_data()*

*RETURNS TRIGGER AS $$*

*BEGIN*

*-- Обновляем данные в таблице purchases при изменении данных в таблице employees*

*UPDATE purchases*

*SET id\_employee = NEW.id*

*WHERE id\_employee = OLD.id;*

*-- Обновляем данные в таблице orders при изменении данных в таблице employees*

*UPDATE orders*

*SET id\_employee = NEW.id*

*WHERE id\_employee = OLD.id;*

*RETURN NEW;*

*END;*

*$$ LANGUAGE plpgsql;*

*CREATE TRIGGER update\_purchase\_and\_order\_employee\_data\_trigger*

*AFTER UPDATE ON employees*

*FOR EACH ROW*

*EXECUTE FUNCTION update\_purchase\_and\_order\_employee\_data();*

Триггер, который отвечает за то, чтобы при удалении товара из таблицы «Товары в покупке», количество удаленного товара восстанавливало остаток в магазине таблице «Товары», а сумма в таблице «Покупка» и «Чек» изменялась автоматически:

*CREATE OR REPLACE FUNCTION update\_on\_product\_removal()*

*RETURNS TRIGGER AS $$*

*BEGIN*

*-- Увеличиваем остаток в магазине*

*UPDATE products*

*SET balance\_in\_store = balance\_in\_store + OLD.quantity*

*WHERE id = OLD.id\_product;*

*-- Обновляем сумму в таблице purchases*

*UPDATE purchases*

*SET total\_amount = COALESCE((SELECT SUM(total\_price) FROM products\_in\_purchase WHERE id\_purchase = purchases.id), 0)*

*WHERE id = OLD.id\_purchase;*

*-- Обновляем сумму в таблице receipts*

*UPDATE receipts*

*SET total\_amount = purchases.total\_amount*

*FROM purchases*

*WHERE receipts.id\_purchase = purchases.id;*

*RETURN OLD;*

*END;*

*$$ LANGUAGE plpgsql;*

*CREATE TRIGGER update\_on\_product\_removal\_trigger*

*AFTER DELETE ON products\_in\_purchase*

*FOR EACH ROW*

*EXECUTE FUNCTION update\_on\_product\_removal();*

Триггер на обновление количества товара на складе при добавлении товара в заказ:

*CREATE OR REPLACE FUNCTION reduce\_quantity\_in\_warehouse\_on\_order()*

*RETURNS TRIGGER AS $$*

*BEGIN*

*-- Обновление количества товара на складе при добавлении товара в заказ*

*UPDATE warehouse*

*SET quantity = quantity - NEW.quantity*

*WHERE id\_product = NEW.id\_product;*

*RETURN NEW;*

*END;*

*$$ LANGUAGE plpgsql;*

*CREATE TRIGGER trigger\_reduce\_quantity\_in\_warehouse\_on\_order*

*BEFORE INSERT ON order\_items*

*FOR EACH ROW*

*EXECUTE FUNCTION reduce\_quantity\_in\_warehouse\_on\_order();*

Триггер который вычисляет итоговую стоимость товаров в заказе на основе цены за единицу товара:

*CREATE OR REPLACE FUNCTION update\_total\_price\_order\_items()*

*RETURNS TRIGGER AS $$*

*BEGIN*

*SELECT price\_unit \* NEW.quantity INTO NEW.total\_price*

*FROM products*

*WHERE id = NEW.id\_product;*

*RETURN NEW;*

*END;*

*$$ LANGUAGE plpgsql;*

*CREATE TRIGGER calculate\_total\_price\_order\_items*

*BEFORE INSERT OR UPDATE OF quantity ON order\_items*

*FOR EACH ROW*

*EXECUTE FUNCTION update\_total\_price\_order\_items();*

Триггер для таблицы "orders", который автоматически подставляет цену, суммируя все "total\_price" в таблице "order\_items" для определенного заказа:

*CREATE OR REPLACE FUNCTION update\_total\_amount\_orders()*

*RETURNS TRIGGER AS $$*

*BEGIN*

*-- Обновляем total\_amount в таблице orders*

*UPDATE orders*

*SET total\_amount = COALESCE((SELECT SUM(total\_price) FROM order\_items WHERE id\_order = orders.id), 0)*

*WHERE id\_order = NEW.id\_order;*

*RETURN NEW;*

*END;*

*$$ LANGUAGE plpgsql;*

*CREATE TRIGGER update\_total\_amount\_orders\_trigger*

*AFTER INSERT OR UPDATE OF total\_price ON order\_items*

*FOR EACH ROW*

*EXECUTE FUNCTION update\_total\_amount\_orders();*

Триггер, который будет автоматически обновлять статус и сумму при создании записи в таблице order\_payments:

*CREATE OR REPLACE FUNCTION update\_order\_status\_and\_amount()*

*RETURNS TRIGGER AS $$*

*BEGIN*

*-- Обновляем статус заказа на "Заказ оплачен и создан"*

*UPDATE orders*

*SET status = 'Заказ оплачен и создан'*

*WHERE id = NEW.id\_order;*

*-- Обновляем сумму в order\_payments из total\_amount таблицы orders*

*UPDATE order\_payments*

*SET amount = (SELECT total\_amount FROM orders WHERE id = NEW.id\_order)*

*WHERE id = NEW.id;*

*RETURN NEW;*

*END;*

*$$ LANGUAGE plpgsql;*

*CREATE TRIGGER update\_order\_status\_and\_amount\_trigger*

*AFTER INSERT ON order\_payments*

*FOR EACH ROW*

*EXECUTE FUNCTION update\_order\_status\_and\_amount();*

Триггер, который отвечает за то, чтобы при удалении товара из таблицы «Товары в заказе», количество удаленного товара восстанавливало остаток на складе в таблице «Склад», а сумма в таблице «Заказ» изменялась автоматически:

*CREATE OR REPLACE FUNCTION update\_quantity\_and\_amount()*

*RETURNS TRIGGER AS $$*

*BEGIN*

*-- Получаем количество и идентификатор товара из удаленной строки в order\_items*

*DECLARE*

*deleted\_quantity INT;*

*deleted\_product\_id INT;*

*BEGIN*

*SELECT quantity, id\_product*

*INTO deleted\_quantity, deleted\_product\_id*

*FROM order\_items*

*WHERE id = OLD.id;*

*-- Увеличиваем остаток на складе*

*UPDATE warehouse*

*SET quantity = quantity + OLD.quantity*

*WHERE id = OLD.id\_product;*

*-- Обновляем общую сумму в таблице orders*

*UPDATE orders*

*SET total\_amount = total\_amount - (deleted\_quantity \* (*

*SELECT price\_unit FROM products WHERE id = deleted\_product\_id*

*))*

*WHERE id = OLD.id\_order;*

*RETURN OLD;*

*END;*

*END;*

*$$ LANGUAGE plpgsql;*

*CREATE TRIGGER update\_quantity\_and\_amount\_trigger*

*AFTER DELETE ON order\_items*

*FOR EACH ROW*

*EXECUTE FUNCTION update\_quantity\_and\_amount();*

## 3.4 Построение физической модели

Физическая модель данных представлена реляционными таблицами, в которых в виде картежей хранится информация.

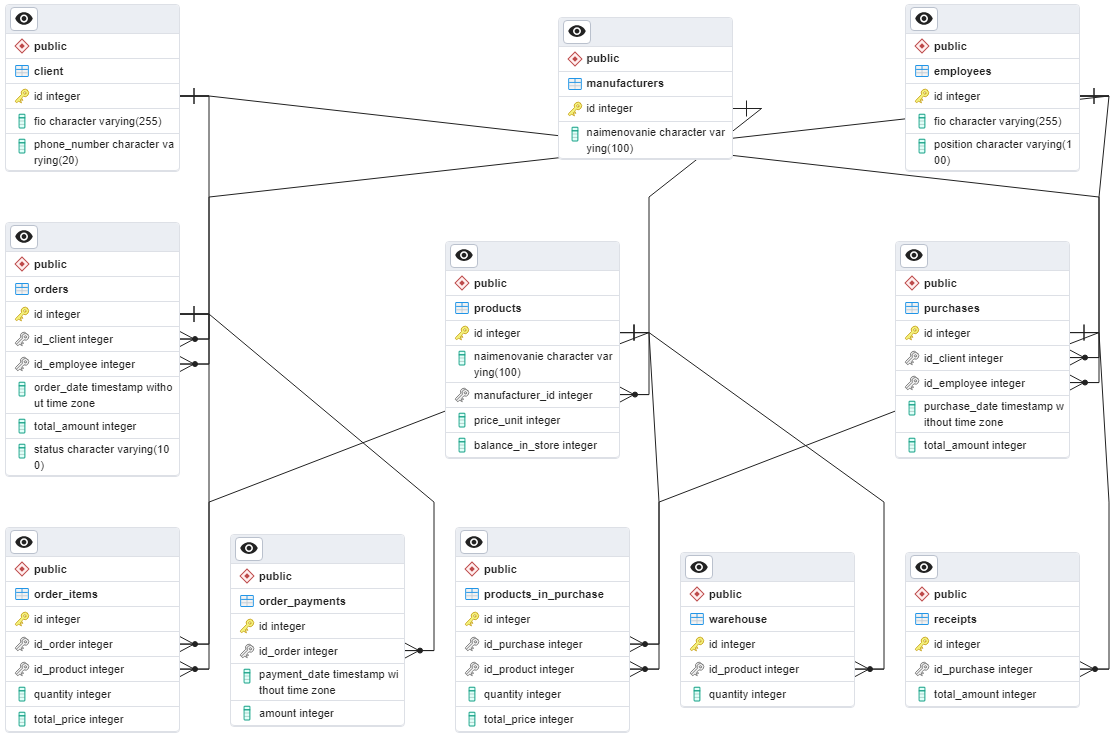


Рисунок 3.3 – Физическая модель базы данных

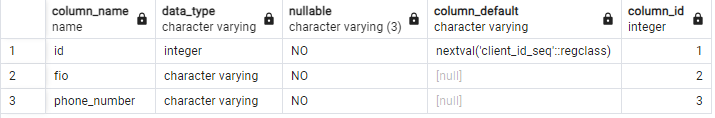


Рисунок 3.4 – Таблица Client «Клиент».



Рисунок 3.4 – Список триггеров таблицы Client.

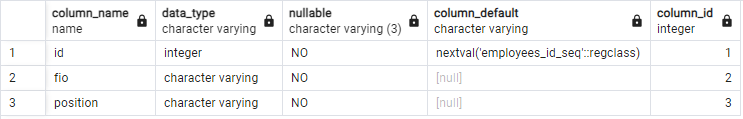


Рисунок 3.6 – Таблица Employees «Сотрудники».



Рисунок 3.7 – Список триггеров таблицы Employees.



Рисунок 3.8 – Таблица Manufacturers «Производители».

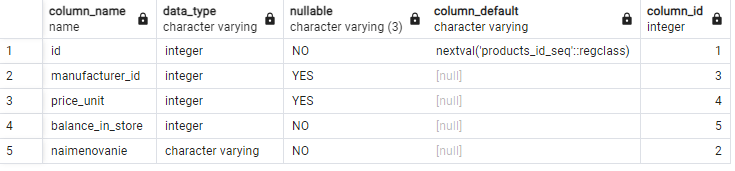


Рисунок 3.9 – Таблица Products «Товары».



Рисунок 3.10 – Список триггеров таблицы Products.

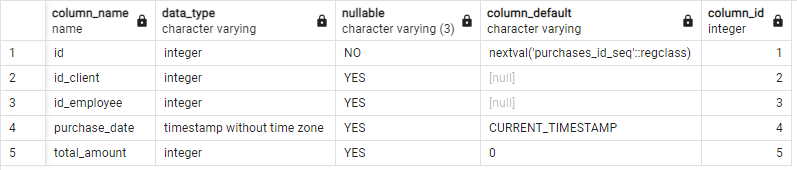


Рисунок 3.11 – Таблица Purchases «Покупка».

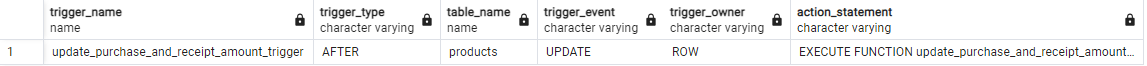


Рисунок 3.12 – Список триггеров таблицы Purchases.

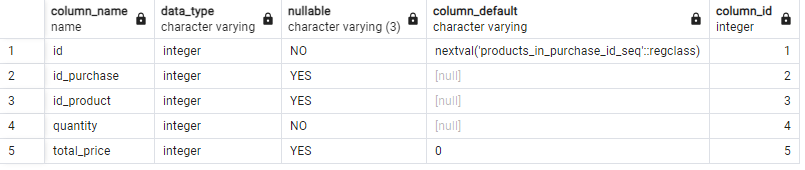


Рисунок 3.13 – Таблица Products\_in\_purchase «Товары в покупке».

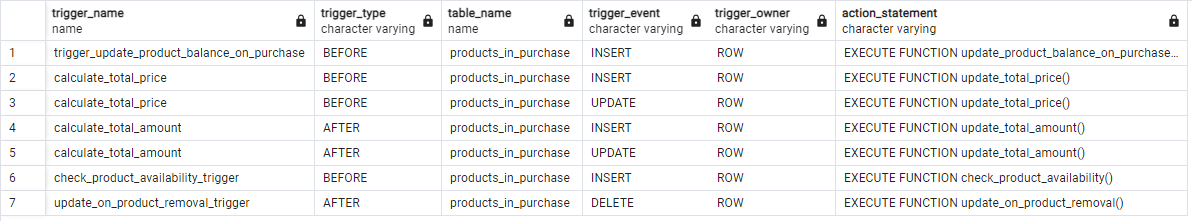


Рисунок 3.13 – Список триггеров таблицы Products\_in\_purchase.

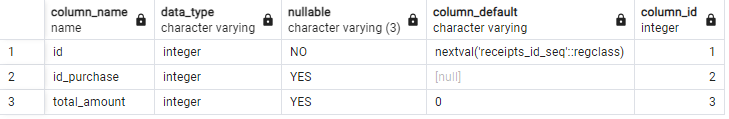


Рисунок 3.14 – Таблица Receipts «Чек».

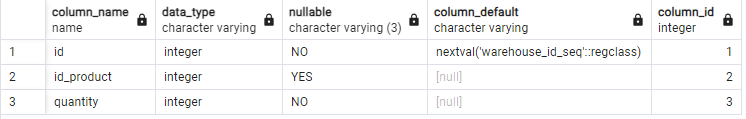


Рисунок 3.15 – Таблица Warehouse «Склад».

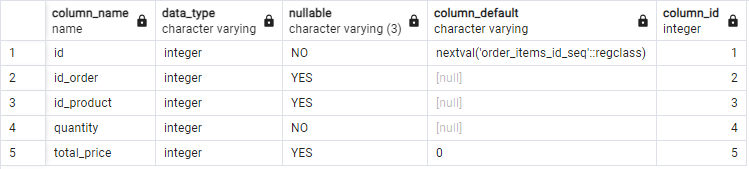


Рисунок 3.16 – Таблица Order\_items «Товары в заказе».

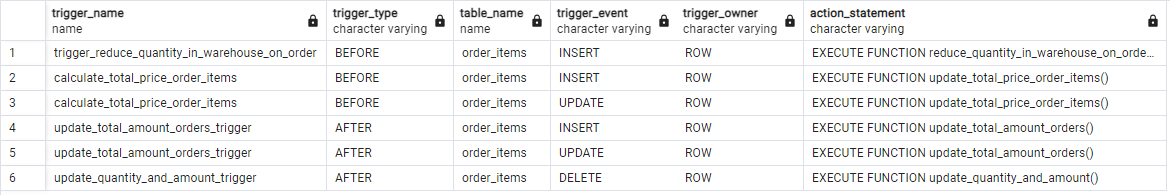


Рисунок 3.17 – Список триггеров таблицы Order\_items.

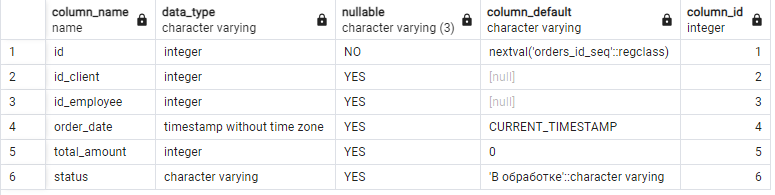


Рисунок 3.18 – Таблица Orders «Заказы».

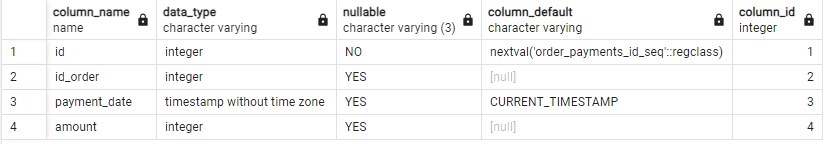


Рисунок 3.19 – Таблица Order\_payments «Оплата заказа».

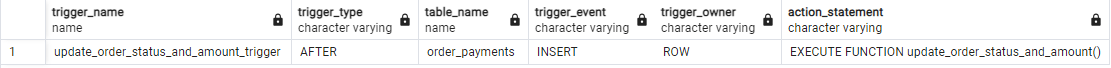


Рисунок 3.20 – Список триггеров таблицы Order\_payments.

Этап физического проектирования требует выделения транзакций – логически законченных последовательностей операций манипулирования данными. Операции манипулирования данными – это чтение (*SELECT*), удаление (*DELETE*), изменение (*UPDATE*), добавление (*INSERT*) данных.

Таблица 3.1. Транзакции, позволяющие реализовать функцию ПОКУПКА ТОВАРА

|  |  |  |
| --- | --- | --- |
| ТРАНЗАКЦИЯ | КОМАНДЫ | *DML-*операции |
| **ПОКУПКА** | Читать данные клиента; | *SELECT \* FROM client;* |
| Читать данные сотрудника; | *SELECT \* FROM employees;* |
| Читать данные о товарах в покупке; | *SELECT \* FROM products\_in\_purchase;* |
| Положить данные о клиенте, сотруднике, а так же указать дату и стоимость в соотвествии с суммой товаров (стоимость вносится автоматически из таблицы «Товары в покупке»); | *INSERT INTO purchases (id\_client, id\_employee) VALUES (id, id);* |
| **ТОВАРЫ В ПОКУПКЕ** | Читать данные покупки; | *SELECT \* FROM purchases;* |
| Читать данные товара; | *SELECT \* FROM products;* |
| Положить данные о покупке, товаре, количестве преобретаемого и стоимости в соответствии с количеством единиц товара (стоимость вносится автоматически из таблицы «Товары»); | *INSERT INTO products\_in\_purchase (id\_purchase, id\_product, quantity) VALUES (id, id, quantity);* |
| **ЧЕК** | Читать номер покупки; | SELECT \* FROM priem; |
| Чек формируется автоматически при создании покупки, стоимость заполняется автоматически из таблицы «Покупка» | *Работа триггера «generate\_receipt\_for\_purchase»* |

В табл. 3.1 перечислены команды, которые составляют каждую из этих транзакций.

## 3.5 Проектирование SQL-запросов

В процессе разработки были созданы SQL-запросы отражающие функционал базы данных.

Ниже представлены SQL-запросы, созданные в процессе разработки информационной системы.

1. Получить список всех клиентов и их суммарные затраты:

*SELECT c.id, c.FIO AS "ФИО", SUM(p.total\_amount) AS "Всего потрачено"*

*FROM client c*

*LEFT JOIN purchases p ON c.id = p.id\_client*

*GROUP BY c.id, c.FIO*

*ORDER BY "Всего потрачено" DESC;*

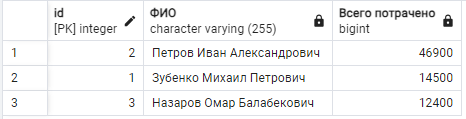


Рисунок 3.20 – Результат запроса 1

1. Найти самый продаваемый товар и его количество:

*SELECT pr.naimenovanie AS "Самый продаваемый товар", SUM(pip.quantity) AS "Всего преобретено"*

*FROM products\_in\_purchase pip*

*JOIN products pr ON pip.id\_product = pr.id*

*GROUP BY pr.naimenovanie*

*ORDER BY "Всего преобретено" DESC*

*LIMIT 1;*

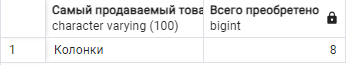


Рисунок 3.22 – Результат запроса 2

1. Получить средний чек за период времени:

*SELECT AVG(total\_amount) AS "Средний чек"*

*FROM purchases*

*WHERE purchase\_date BETWEEN '2023-01-01' AND '2024-01-07';*

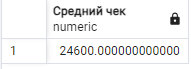


Рисунок 3.23 – Результат запроса 3

1. Получить список сотрудников и количество продаж, суммарная выручка:

*SELECT e.id, e.FIO AS "ФИО", COUNT(p.id) AS "Всего продано", SUM(p.total\_amount) AS "Суммарная выручка"*

*FROM employees e*

*LEFT JOIN purchases p ON e.id = p.id\_employee*

*GROUP BY e.id, e.FIO*

*ORDER BY "Суммарная выручка" DESC;*

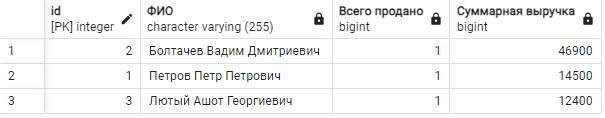


Рисунок 3.24 – Результат запроса 4

1. Получить список производителей и среднюю цену их товаров:

*SELECT m.naimenovanie AS "Производители", AVG(pr.price\_unit) AS "Средняя цена товара"*

*FROM manufacturers m*

*JOIN products pr ON m.id = pr.manufacturer\_id*

*GROUP BY m.naimenovanie;*



Рисунок 3.25 – Результат запроса 5

1. Получить информацию о продуктах, у которых остаток в магазине больше 10:

*SELECT \**

*FROM products*

*WHERE balance\_in\_store > 10;*



Рисунок 3.26 – Результат запроса 6

1. Получить список покупок с деталями товаров и суммой:

*SELECT pu.id, pu.purchase\_date AS "Дата покупки", c.FIO AS "ФИО клиента", e.FIO AS "ФИО сотрудника",*

*pr.naimenovanie AS "Наименование товара", pip.quantity AS "Количество", pip.total\_price "Итоговая цена"*

*FROM purchases pu*

*JOIN client c ON pu.id\_client = c.id*

*JOIN employees e ON pu.id\_employee = e.id*

*JOIN products\_in\_purchase pip ON pu.id = pip.id\_purchase*

*JOIN products pr ON pip.id\_product = pr.id;*

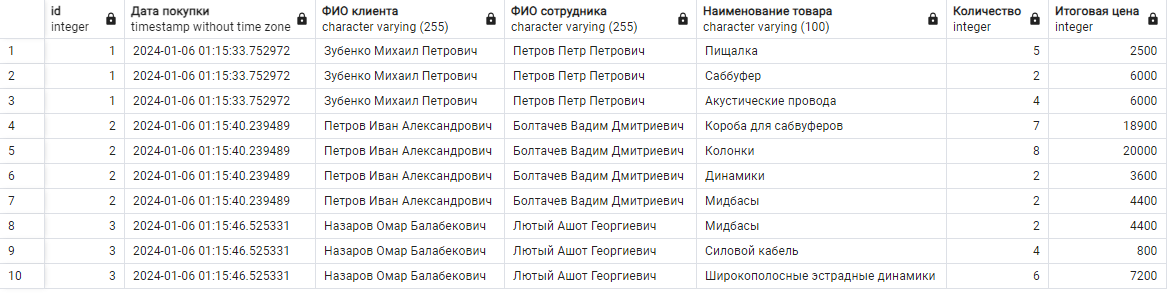


Рисунок 3.27 – Результат запроса 7

1. Получить среднюю стоимость покупки для каждого клиента:

*SELECT c.id, c.FIO, AVG(p.total\_amount) AS average\_purchase\_cost*

*FROM client c*

*JOIN purchases p ON c.id = p.id\_client*

*GROUP BY c.id, c.FIO;*

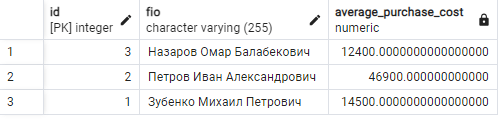


Рисунок 3.28 – Результат запроса 8

1. Получить общую стоимость товаров у каждого производителя:

*SELECT m.naimenovanie AS "Производители", SUM(pr.price\_unit \* pr.balance\_in\_store) AS "Общая стоимость"*

*FROM manufacturers m*

*JOIN products pr ON m.id = pr.manufacturer\_id*

*GROUP BY m.naimenovanie;*

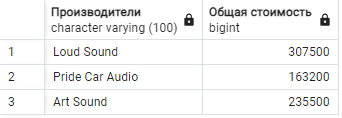


Рисунок 3.29 – Результат запроса 9

1. Получить список товаров, у которых цена выше средней цены по всем товарам:

*SELECT naimenovanie AS "Наименование", price\_unit "Цена"*

*FROM products*

*WHERE price\_unit > (SELECT AVG(price\_unit) FROM products);*

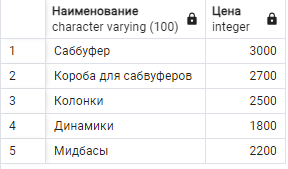


Рисунок 3.30 – Результат запроса 10