МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ»
Кафедра IT-инжиниринг

ДОПУЩЕН К ЗАЩИТЕ
Заведующий кафедрой
PhD, доцент
_____ Т.С. Картбаев
«____» _____ 2018 г.

ДИПЛОМНЫЙ ПРОЕКТ

На тему: Разработка базы данных и программного обеспечения для производственного склада

Специальность 5В070400 – «Вычислительная техника и программное обеспечение»

Выполнил Хашимжан Д.А Группа ВТ-14-2
Научный руководитель ст. преп. Рахимжанова З.М.

Консультанты:
по экономической части: к.э.н., профессор _____ Ж.Г. Аренбаева
«22» мая 2018 г.

по безопасности жизнедеятельности: ст. преп. _____ А.А. Абикенова
«21» 05 2018 г.

по применению
вычислительной техники: ст. преп. _____ А.М. Рамазанова
«22» 05 2018 г.

Нормоконтролер: PhD, ст. преп. _____ Ж. Бидахмет
«30» 05 2018 г.

Рецензент: PhD, ассистент _____ О.Б. Баймуратов
«____» _____ 2018 г.

Алматы 2018

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ»

Институт систем управления и информационных технологий

Кафедра IT-инжиниринг

Специальность 5В070400 – «Вычислительная техника и программное обеспечение»

ЗАДАНИЕ
на выполнение дипломного проекта

Студенту Хашимжан Дильяру Арманұлы

Тема работы: Разработка базы данных и программного обеспечения для производственного склада

Утверждена приказом по университету №155 от «23» октября 2017 г.

Срок сдачи законченной работы «1» июня 2018 г.

Исходные данные к работе (требуемые параметры результатов исследования (проектирования) и исходные данные объекта): Oracle – среда разработки, C# - язык программирования, SQL Developer – графический интерфейс, Visual Studio – интегрированная среда разработки.

Перечень вопросов, подлежащих разработке в дипломном проекте, или краткое содержание дипломного проекта:

а) разработка базы данных для склада;
б) методы разработки базы данных;
в) сравнение различных баз данных
г) вопросы безопасности жизнедеятельности и охраны труда;
д) экономическая эффективность работ по стандартизации.

Перечень графического материала (с точным указанием обязательных чертежей): представлены 27 таблиц, 30 иллюстрации.

Основная рекомендуемая литература:

1 Роб П., Коронел К. Системы баз данных: проектирование, реализация и управление. – 5-е изд. – Спб.: БХВ-Петербург, 2004. – 1040 с.

2 Джейсон Прайс, Майк Гандэрлой. Visual C# .NET. Полное руководство. – КОРОНА принт, 2004. – 960 с.

3 Баженова И. Ю. C++ && Visual Studio NET. Самоучитель программиста.– М.: КУДИЦ-ОБРАЗ, 2003. – 448с.

4 Шмуллер Д. Освой самостоятельно UML за 24 часа. [Текст]: М.: Издательский дом «Вильямс», 2005. – 416 с.

Консультации по работе с указанием относящихся к ним разделов работы

Раздел	Консультант	Сроки	Подпись
Экономическая часть	Аренбаева Ж.Г.	30.03.2018-22.05.2018	
Безопасности жизнедеятельности	Абикенова А.А.	30.03.2018-21.05.2018	
Программная часть	Рамазанова А.М.	05.05.2018-22.05.2018	
Нормоконтролер	Бидахмет Ж.	23.05.2018-31.05.2018	

График подготовки дипломного проекта

Наименование разделов, перечень разрабатываемых вопросов	Сроки представления научному руководителю	Примечание
Анализ предметной области	6.02.2018-21.02.2018	выполнено
Выбор программно-аппаратных средств	22.02.2018-19.03.2018	выполнено
Разработка структуры базы данных	20.03.2018-09.04.2018	выполнено
Реализация базы данных в графическом интерфейсе	09.04.2018-27.04.2018	выполнено
Безопасность жизнедеятельности	30.03.2018-21.05.2018	выполнено
Экономическое обоснование проекта	30.03.2018-22.05.2018	выполнено

Дата выдачи задания «25» октября 2017 г.

Заведующий кафедрой _____ Т.С. Картбаев

Научный руководитель работы _____ З.М. Рахимжанова

Задание принял к исполнению студент _____ Д.А. Хашимжан

Содержание

Введение

Склад описывает объект, предназначенный для хранения товаров. В электронной торговле склады используются в основном для хранения предметов на складе, чтобы убедиться, что короткие сроки доставки могут быть выполнены.

Складирование и все, что с ним связано, являются частью сложной отрасли, известной как управление логистикой. Логистика включает в себя закупки, управление запасами и распределение. Он подпадает под зонтик цепочки поставок, который также включает в себя разработку продуктов, маркетинг, продажи и другие связанные с продуктом дисциплины.

Каждая организация имеет информацию о том, что она должна хранить и удовлетворять требованиям. Например, корпорация должна собирать и вести записи о людских ресурсах для своих сотрудников. Эта информация должна быть доступна тем, кто в ней нуждается. Информационная система представляет собой формальную систему для хранения и обработки информации.

Информационной системой может быть набор картонных ящиков, содержащих папки манила, наряду с правилами хранения и извлечения папок. Однако сегодня большинство компаний используют базу данных для автоматизации своих информационных систем.

База данных представляет собой набор информации, организованной таким образом, что она легкодоступна, управляется и обновляется.

Данные упорядочены и индексируются для непосредственного облегчения поиска какой-либо информации. Данные изменяются, расширяются и удаляются по мере прибавления новой информации. Базы данных обрабатывают трудовые нагрузки для образования и обновления самих себя, запроса данных, которые они содержат, и запуска приложений против него.

Целью базы данных является сбор, хранение и получение соответствующей информации для использования приложений баз данных.

Система управления базами данных (СУБД) - это программное обеспечение, которое выполняет такие функции, как хранить, организовать и извлекать данные. Как правило, СУБД имеет следующие элементы:

– код ядра. Этот код управляет памятью и хранилищем для СУБД;

– магазин метаданных. Этот контейнер обычно называют словарем данных;

– язык для запросов. Этот язык позволяет приложениям получать доступ к данным.

Приложение базы данных - это программное обеспечение, которое взаимодействует с базой данных для доступа к данным и управления ими.

Первое поколение систем управления базами данных включает в себя следующие типы:

– иерархический. Иерархическая база данных выполняет данные в виде древовидной структуры. Запись каждого родителя имеет одну или несколько дочерних записей, таких как структура файловой системы;

– сеть. Сетевая база данных похожа на иерархическую базу данных, за исключением записей, которые много-ко-многим, а не один-ко-многим.

Более ранние системы управления базами данных хранят данные в жестких, предопределенных отношениях. Поскольку не существовал язык определения данных, трудно изменить структуру данных. Кроме того, в этих системах нет простого языка запросов, который затруднил разрабатывать приложения.

1 Планирование и начальная разработка

1.1 Анализ предметной области

Когда дело доходит до настройки больших, сложных сред хранилищ данных, похоже, имеется множество доступных вариантов. Однако, несмотря на все инструменты настройки и программные функции, консультанты и учебные материалы, полностью оптимизируя складскую среду, по-прежнему чрезвычайно длинны, усеянные оговорками и блокпостами. Существует несколько процессов для анализа, несколько задействованных команд и, как правило, огромное количество данных, поступающих на склад и через него. Вся эта деятельность по необходимости будет сосредоточена на базах данных источника продукции, а также на складах производственных данных.

Для обеспечения успеха в сложной складской экосистеме служит создание базы данных склада. С помощью БД архитекторы данных и разработчики могут тестировать новые и улучшенные сценарии загрузки. Администраторы баз данных могут работать с группами разработчиков для реорганизации архитектуры и настройки процессов и запросов базы данных. Инженеры могут тщательно протестировать изменения, чтобы обеспечить надежную среду. А аналитики, конечные пользователи, руководители и группы отчетности могут получать нужные им данные, когда они им нужны, не влияя на других пользователей или процессы.

Базы данных используются совместно с графическим пользовательским интерфейсом в отделе, где сотрудники хранят данные и редактируют базу данных. Работа этого приложения:

– упрощает обработку данных;

– целостность данных;

– удобство и сокращение времени на удаление информации;

– удаление старых методов процедур производства в работе.

Анализ области деятельности и предметной области проведен:

– на основе собранных данных и информации о предстоящих планах компании;

– основываясь на опыте предыдущих практик создания баз данных для разных типов бизнеса.

– основываясь на важности управления базой данных для компании;

– на основе ожидаемых изменений в производительности компании и ее сотрудников.

В результате анализ должен привести к четкому пониманию моделей для поэтапной рабочей силы для достижения конечных результатов. Только после анализа вы можете понять, сколько данных вам нужно создать. Четкая структура его деятельности и структура исследования выявят все бизнес-потребности продаж и отделов в области автоматизации и кадрового обеспечения.

1.2 Постановка задачи

База данных проектируется для производственного склада. БД должна содержать информацию:

– о складе;
– о персонале склада;
– о заказах;
– о производителях;
– о товарах.

С помощью БД можно отслеживать товарооборот, работоспособность того или иного сотрудника.

В связи с огромным количеством разнообразных товаров, склад разделен на несколько крупных отделов:

– ноутбуки и ПК;
– смартфоны;
– бытовая техника.

В отделах информация о каждом товаре предоставлена в развернутом виде.

База данных дипломного проекта разрабатывается вместе с визуальным интефейсом, где сотрудник ведущий контроль может с легкостью находить необходимый товар, а также добавлять и удалять сотрудников склада.

1.3 Задачи проектирования базы данных

Когда программное обеспечение продвигается вперед, целью является упрощение рабочих процессов компании, сокращение времени, необходимого для обработки подпрограмм HR, и упрощение отдела продаж путем ввода определенной базы данных в базу данных. Интегрируя графический интерфейс в саму базу данных, введение сотрудника или группы продуктов становится более простой задачей. В ежеквартальном анализе продаж отдел продаж может запросить информацию о желаемом продукте.

Основные задачи выбираются как наиболее распространенная методология для рабочего процесса этих отделов, и выполняется работа по определению ступенчатой структуризации этого процесса, затем решение выполняется в среде разработки, которая отвечает за осуществление.

1.4 UML диаграммы

Unified Modeling Language (Унифицированный Язык Программирования) является типовым языком визуального моделирования, предназначен для применения в:

- моделирование бизнес-процессов и связанных с ними процессов;
- анализе, проектировании и внедрении программных систем.

UML - язык бизнес-аналитиков, архитекторов и разработчиков программного обеспечения для представления, проектирования и документирования существующих или новых бизнес-процессов, структуры и поведения систем программного стиля.

UML можно применять для различных приложений (например, банковское дело, финансы, интернет, аэрокосмическая промышленность, здравоохранение и т. д.). Он может использоваться во всех основных технологиях разработки и программных компонентах для реализации различных платформ (например, J2EE, .NET).

UML - это стандартный язык моделирования, а не процесс разработки программного обеспечения. Процесс:

- дает указания в порядке команды;
- определяет, какие сущности должны быть построены;
- управляет деятельностью отдельных разработчиков и команды в целом;
- обеспечивает критерии для отслеживания и измерения продуктов и деятельности по проекту.

UML намеренно отделен от процесса и может использоваться в контексте различных процессов. Однако он лучше всего подходит для процессов обучения в повторяющемся и постепенном развитии.

UML является неполным и не полностью визуальным. Учитывая некоторую UML-диаграмму, мы не можем точно понять изображенную часть или поведение системы только на диаграмме. Некоторая информация может быть намеренно опущена на диаграмме, некоторая информация, представленная на диаграмме, может иметь разные интерпретации, а некоторые понятия UML вообще не имеют графической нотации, поэтому нет способа изобразить на диаграммах.

1.4.1 Диаграмма классов

Диаграмма классов представляет собой статическую диаграмму. Это статический вид приложения. Он не только описывает, иллюстрирует, классифицирует различные аспекты системы, но также создает исполняемый код для программного приложения.

Диаграмма классов описывает характеристики и работу класса и ограничения, налагаемые на систему. Диаграммы классов часто используются при моделировании объектно-ориентированных систем, поскольку они представляют собой только UML-диаграммы, которые можно просматривать непосредственно в объектно-ориентированных языках.

Диаграмма классов показывает ряд классов, интерфейсов, сопоставлений, совместной работы и барьеров. Он также известен как диаграмма структуры.

Цель диаграммы классов - имитировать статический внешний вид приложения. Диаграммы классов - это единственные диаграммы, которые

могут быть непосредственно сопоставлены с объектно-ориентированными языками и поэтому широко используются во время построения.

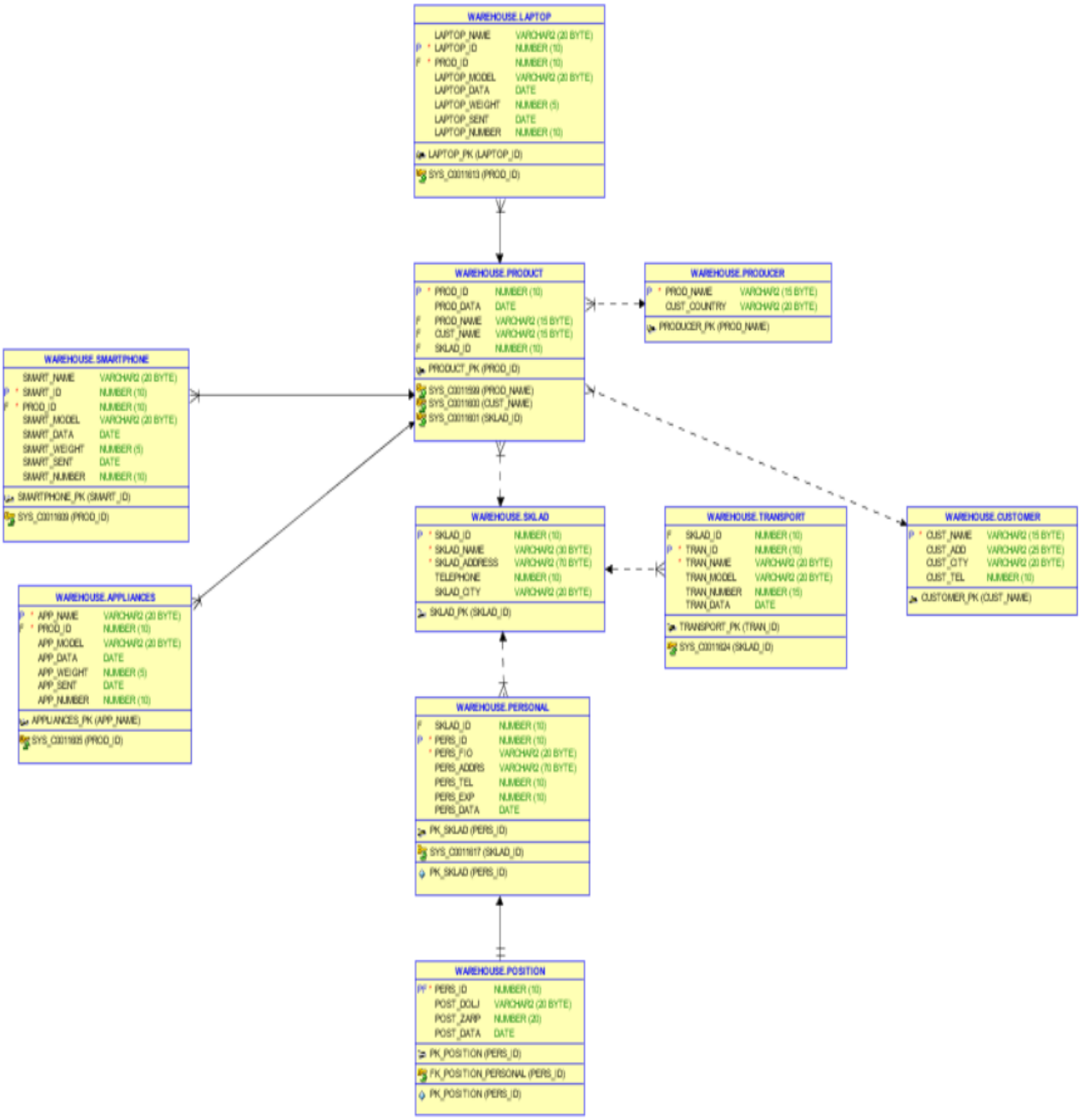


Рисунок 1.1. – Диаграмма классов

1.4.2 Диаграмма развертывания

Диаграммы развертывания используются для иллюстрации топологии физических компонентов системы, в которой развертываются программные компоненты.

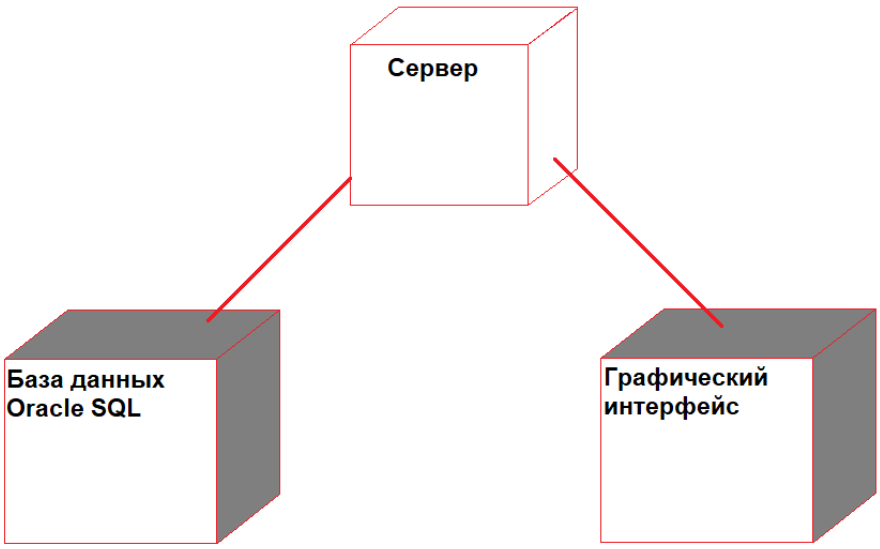


Рисунок 1.2. – Диаграмма развертывания

Диаграммы развертывания используются для описания статического типа развертывания системы. Диаграммы развертывания состоят из узлов и их связей.

Сам термин «Развертывание» иллюстрирует цель диаграммы. Диаграммы развертывания используются для описания аппаратных компонентов, в которых были развернуты компоненты программного обеспечения. Диаграммы компонентов и диаграммы развертывания тесно связаны.

Компонентные диаграммы используются для описания компонентов, а диаграммы развертывания показывают, как они развертываются на аппаратном уровне.

1.4.3 Диаграмма прецедентов

Диаграммы прецедентов используются для сбора системных требований, включая внутренние и внешние воздействия. Эти требования в основном необходимы для дизайна. Непосредственно, когда система оценивается для сбора своей функциональных возможностей, примеры приложения разрабатываются и идентифицируются участниками.

Когда начальная задача завершена, диаграммы случайных ситуаций моделируются для представления внешнего вида.

Целями диаграмм прецедентов можно назвать следующие:

- сбор системных требований;
- используется для захвата внешнего вида системы;
- определяет внешние и внутренние факторы, которые влияют на систему;
- показывает взаимосвязь между требованиями, являются субъектами.

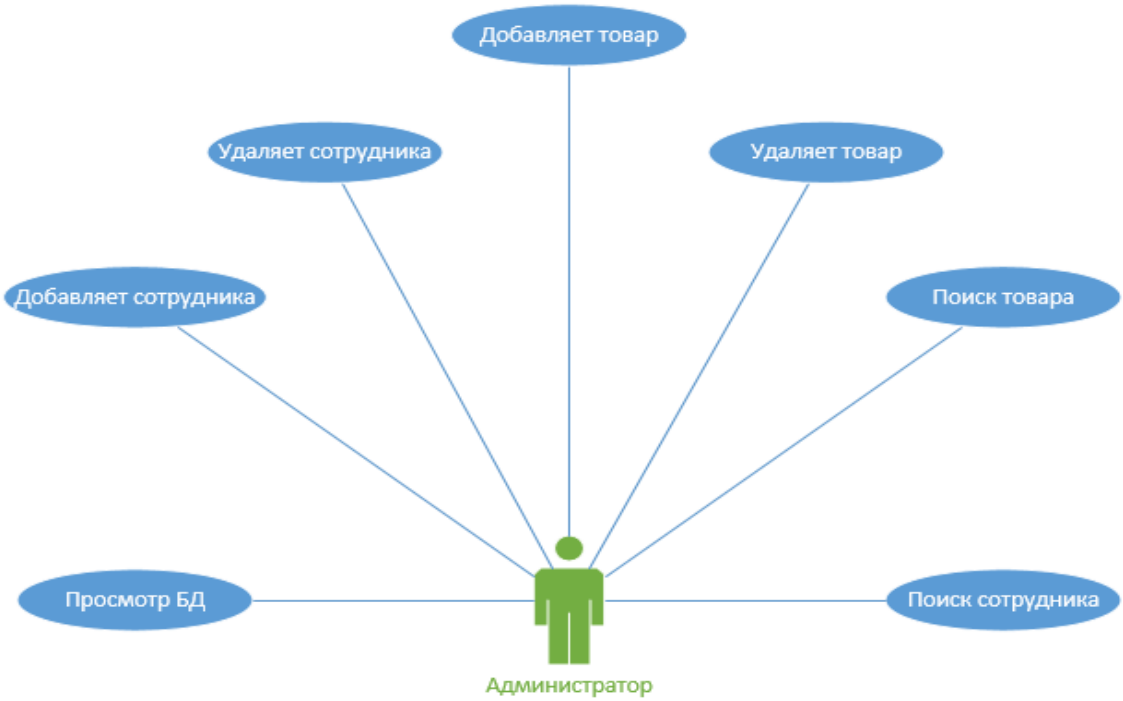


Рисунок 1.3. – Диаграмма прецедентов

2 Выбор программных обеспечений

2.1 Microsoft Visio

Microsoft Visio - это программное обеспечение для рисования различных диаграмм. К ним относятся блок-схемы, организационные диаграммы, чертежи зданий, планы этажей, диаграммы потоков данных, диаграммы технологических процессов, моделирование бизнес-процессов, диаграммы плавательных карт, 3D-карты и т.д. Это продукт Microsoft, продаваемый в качестве как дополнение к MS Office.

Рисунок 2.1. – Категории шаблонов представленных в Microsoft Visio

Visio включает в себя большую библиотеку форм / значков, используемых в десятках типов диаграмм. Эти символы представляют собой специализированные диаграммы, такими как диаграммы технологических процессов, моделирование бизнес-процессов, диаграммы потоков данных и многие другие. Они широко используются в различных областях для различных целей. Вот несколько примеров:

– в каждом поле: А блок - схемы, которые может принимать различные формы, может использоваться для документирования и анализа процесса. Стандартизация процесса передового опыта и качества; передача учебного процесса или понимание других организационных единиц; и выявлять и улучшать узкие места, увольнения и ненужные шаги в этом процессе.

С разработкой программного обеспечения и бизнес-анализом: диаграммы потоков данных (DFD) могут обеспечить специальную стратегию технического проектирования, при этом больше исследований проводится заранее, чтобы изменить кодирование. Business Intelligence использует DFD для изучения существующих систем и поиска превосходства.

Диаграмма процесса может показывать шаги, которые могут быть упущены или не полностью поняты.

Бизнес-моделирование и регистрация бизнес-процессов (BPMN) нацеливают посетителей и других заинтересованных лиц в бизнес-процесс, чтобы получить представление о простой и понятной визуальной презентации шагов. На более активном уровне это для людей, которые внедряют этот процесс и обеспечивают достаточную детализацию для обеспечения точной реализации.

В области химического машиностроения или технологического процесса: Технологическая схема потока (PFD) - это тип блок-схемы, который показывает взаимосвязь между основными компонентами на промышленном предприятии. Диаграммы могут доставлять, анализировать, проверять или лучше отражать документы.

2.2 Oracle 10g

Oracle Database 10 *g* Express Edition (Oracle Database XE) - бесплатная версия наиболее эффективной в мире реляционной базы данных. Oracle Database XE прост в установке, легко управляется и легко развивается.

В Oracle Database XE используется интуитивно понятный интерфейс на основе браузера:

- управление базой данных;
- создание таблиц, представлений и других объектов базы данных;
- импортирование, экспортирование и просматривание данных таблицы;
- реализация запросов и SQL-скриптов;
- создание отчетов.

Требованием многопользовательской РСУБД является контроль параллелизма, который является одновременным доступом одних и тех же данных несколькими пользователями. Без контроля параллелизма пользователи могут неправильно изменять данные, компрометируя целостность данных. Например, один пользователь может обновить строку, а другой пользователь одновременно обновит ее.

Когда несколько пользователей получают доступ к тем же данным, один из способов управления параллелизмом - заставить пользователей ждать. Однако целью СУБД является сокращение времени ожидания, чтобы оно не существовало или было менее важным. Все инструкции SQL изменяют, что данные должны проходить как можно меньше помех. Следует избегать деструктивного взаимодействия, которые не обновляют данные правильно или не меняют структуру данных.

Oracle Database использует блокировки для управления одновременным доступом к данным. Блокировка - это механизм, который предотвращает деструктивное взаимодействие транзакций доступа с общим ресурсом. Блокировки помогают обеспечить целостность данных, обеспечивая при этом максимальный доступ к данным.

В Oracle Database каждый пользователь должен видеть последовательное представление данных, включая видимые изменения, сделанные собственными транзакциями пользователя и транзакциями, совершаемыми другими пользователями. Например, база данных не должна допускать неправильного считывания, когда транзакция обнаруживает несоответствующие изменения, сделанные другой параллельной транзакцией.

База данных Oracle всегда обеспечивает постоянный уровень чтения на уровне команд, гарантируя, что данные, возвращаемые запросом, собираются и сверяются за раз. В зависимости от уровня разделения транзакций это время открытия отчета или начало времени транзакции.

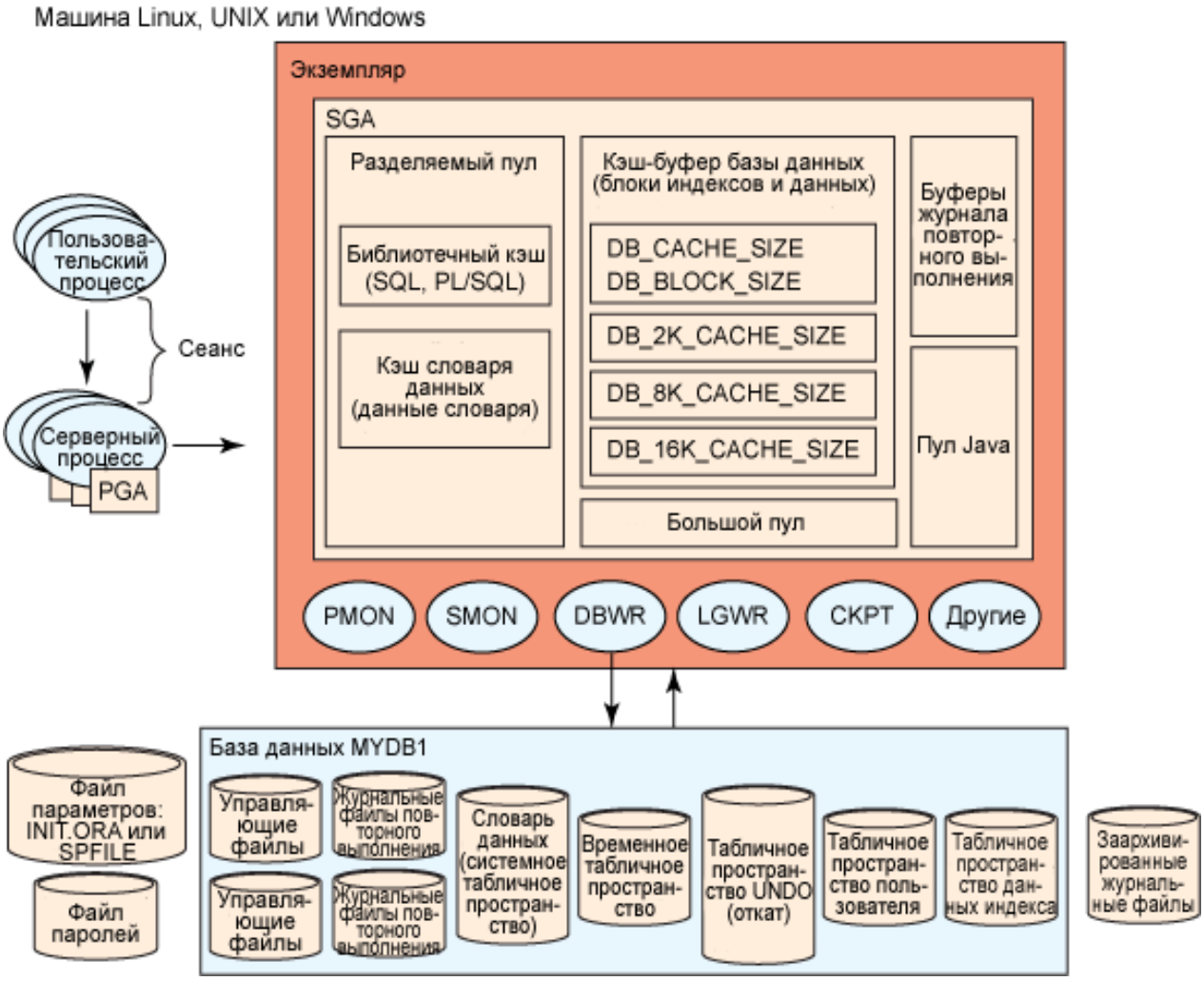


Рисунок 2.2. – Архитектура Oracle

2.2.1 Реляционная модель

Реляционная база данных - это база данных, которая соответствует реляционной модели. Реляционная модель имеет следующие основные аспекты:

– структуры. Определенные объекты, которые хранят или получают доступ к данным базы данных;

– операции. При четко определенных действиях приложения могут манипулировать структурами данных и базами данных;

– правила целостности. Правила контроля целостности в структурах данных и базах данных.

Реляционная база данных хранит данные в нескольких простых отношениях. Связь состоит из множества кортежей. Tuple (кортеж, строка) - это неупорядоченный набор значений атрибутов.

Таблица представляет собой двумерное представление отношения в виде строк (кортежей) и столбцов (атрибутов). Каждая строка в таблице имеет одинаковый набор столбцов. Реляционная база данных - это база данных, которая хранит данные в отношениях (таблицах). Реляционная база данных может быть, например. Например, храните информацию о сотрудниках сотрудников, таблицах отделов и платежных ведомостях.

2.2.2 Реляционная система управления базами данных (РСУБД)

Реляционная модель является основой для системы управления реляционными базами данных (РСУБД). Важно отметить, что СУБД перемещает данные в базу данных, сохраняет данные и извлекает данные, чтобы приложения могли ее манипулировать. СУБД определяет следующие типы операций:

– логические операции. В этом случае приложение определяет, какой контент требуется. Например, приложение запрашивает имя сотрудника или добавляет запись сотрудника в таблицу;

– физические операции. В этом случае РСУБД определяет, как была выполнена и выполнена операция. Например, после запроса консультации в качестве таблицы база данных может использовать индекс для поиска строк петиции, чтения данных памяти и т. Д., Чтобы выполнить следствие перед возвратом результата пользователю. РСУБД хранит и извлекает данные, чтобы физические операции были прозрачными в приложениях баз данных.

База данных Oracle представляет собой реляционную базу данных, которая реализует объектно-ориентированные функции, такие как пользовательские типы, наследование и полиморфизм, систему управления, вызываемую через объектно-реляционную базу данных (ORDBMS). Oracle Database расширяет реляционную модель до объектно-реляционной модели, которая позволяет хранить сложные бизнес-модели в реляционной базе данных.

2.2.3 Язык структурированных запросов (SQL)

SQL - это декларативный, основанный на наборах язык, который обеспечивает интерфейс к СУБД. Например, база данных Oracle. В отличие от языковых методов, таких как C, который описывает, как это сделать, SQL не является методом и описывает, что вам нужно делать. Пользователи ссылаются на желаемые результаты (например, имена текущих сотрудников), а не на то, как они должны быть получены. SQL является стандартным языком реляционных баз данных ANSI.

Все операции с данными в базе данных Oracle выполняются с использованием операторов SQL. Например, вы используете SQL для создания таблиц и запросов и изменения данных в таблицах. Операторы SQL можно рассматривать как очень простую, но мощную компьютерную программу или инструкцию. Оператор SQL представляет собой текстовую строку SQL (рисунок 2.1).

```
SELECT first_name, last_name FROM employees;
```

Рисунок 2.3. – Запрос для получения фамилии и имени из таблицы employees

Операторы SQL позволяют выполнять следующие задачи:

- данные запроса;
- вставить, обновить и удалить строки в таблице;
- создавать, заменять, изменять и удалять элементы;
- контролировать доступ к базе данных и ее объектам;
- совместимость базы данных и целостность.

SQL объединяет предыдущие задачи на одном согласованном языке. Oracle SQL – это реализация стандарта ANSI. Oracle SQL поддерживает множество функций, помимо стандартного SQL.

2.2.4 PL/SQL и Java

PL/SQL является процедурным расширением для Oracle SQL. PL/SQL интегрирован с Oracle Database, позволяя вам использовать все операторы, функции и типы данных Oracle Database SQL. Вы можете использовать PL/SQL для управления потоком программы SQL, использовать переменные и писать методы обработки ошибок.

Основным преимуществом PL/SQL является возможность хранения логики приложения в самой базе данных. Метод или функция PL / SQL - это объект схемы, который состоит из набора операторов SQL и других черновиков PL / SQL, агрегируется, хранится в базе данных и работает как единое целое для решения конкретной проблемы или выполнения одного подключенной линии.

Основным преимуществом серверного программирования является то, что встроенные функции могут быть развернуты в любом месте.

База данных Oracle также может хранить программные модули, написанные на Java. Хранимая процедура Java - это метод Java, который публикуется в SQL и хранится в базе данных для общего использования. Вы можете вызывать существующие PL/SQL-программы из Java и Java-программ из PL/SQL.

2.2.5 Структуры хранения базы данных

Важной задачей реляционной базы данных является хранение данных.

Физические структуры базы данных - это файлы, которые хранят данные. При выполнении команды SQL CREATE DATABASE создаются следующие файлы:

– дата файлы. Каждая база данных Oracle имеет один или несколько файлов физических данных, содержащих все данные базы данных. Данные логических структур баз данных, таких как таблицы и индексы, физически хранятся в файлах данных.

– управляющие файлы. Каждая база данных Oracle имеет файл управления. Файл управления содержит метаданные, определяющие физическую структуру базы данных, включая имя базы данных, а также имена и местоположения файлов базы данных.

– унлайн-файлы журналов повтора. Каждая база данных Oracle имеет онлайн - журнал повторения, который представляет собой набор из двух или более онлайн журнальных файлов. Онлайн - журнал повторения состоит из записей повторного использования (также называемых повторными записями), в которых записаны все изменения, внесенные в данные.

Многие другие файлы необходимы для функционирования сервера базы данных Oracle. Эти файлы включают файлы параметров и файлы диагностики. Резервные файлы и архивированные файлы журналов повторного использования являются автономными, ценными для резервного копирования и восстановления.

2.2.6 Таблицы и таблицы кластеров

Схема базы данных представляет собой логический контейнер для структур данных, называемых элементами. Примерами объектов схемы являются таблицы и индексы. Эта мера создается и обрабатывается в SQL.

У пользователя базы данных есть пароль и некоторые базы данных с привилегиями. Каждый пользователь имеет схему с тем же именем, что и пользователь. Схема содержит данные для пользователя, владеющий схемой, например, время пользователя имеет почасовое расписание, которое содержит объект предложения и таблицу сотрудников. В производственной базе данных

владельцем собственности обычно является приложение базы данных, а не лицо.

Внутри схемы каждый объект схемы определенного типа имеет уникальное имя. Например, hr.employees относятся к таблице персонала в схеме hr. На рисунке 2.4 показаны имена объектов схемы hr и объектов в схеме hr.

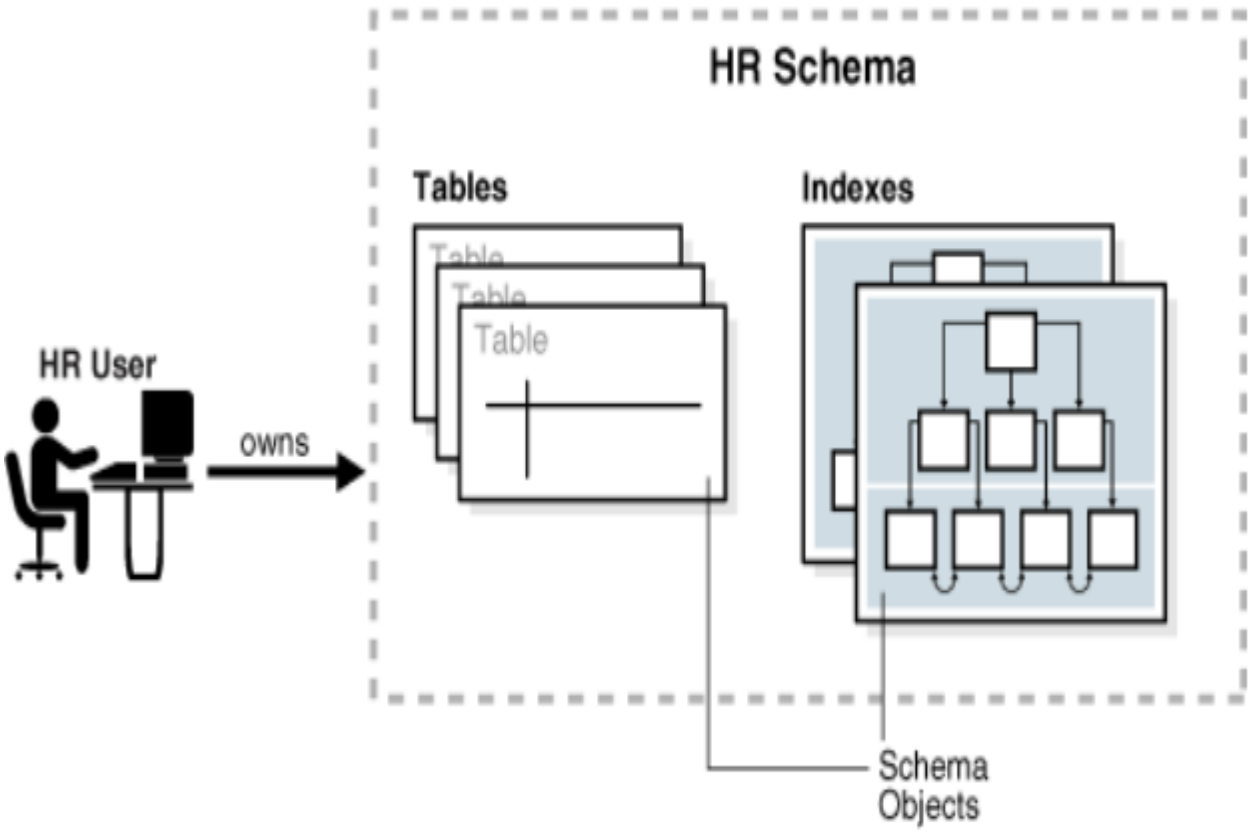


Рисунок 2.4. – Имена hr объектов схемы

Важнейшими мерами реляционной базы данных являются таблицы. Таблица сохраняет данные в столбцах.

Oracle SQL позволяет создавать и манипулировать многими другими вещами в предложении, в том числе:

– индексы. Индексы представляют собой объекты схемы, которые содержат таблицу для каждой индексированной строки в таблице или кластере таблиц, которая обеспечивает прямой и быстрый доступ к строкам. База данных Oracle поддерживает какой-то индекс. Индексированная таблица - это таблица, в которой хранятся данные в структуре индекса;

– разделы. Разделы являются частью больших таблиц и индексируются. Каждый раздел имеет свое имя и может иметь свои собственные свойства памяти;

– представления. Представления представляют собой персонализированные представления данных по одной или нескольким таблицам или дру-

гим. Вы можете думать о них как о сохраненных запросах. Презентации не имеют данных;

– последовательности. Множество пользователей, созданных пользователем, могут совместно использовать целые числа для генерации целых чисел. Часто порядок используется для генерации значений для первичного ключа;

– габаритные размеры. Размерность относится к родительским и дочерним отношениям между столбцами наборов, тогда как столбцы во всех столбцах должны быть из одной таблицы. Размеры обычно используются для категоризации данных, таких как клиенты, продукты и время;

– синоним. Синоним - это псевдоним для другой меры. Поскольку синоним - это только псевдоним, ему не нужна никакая память, кроме ее значения в словаре данных;

– PL / SQL подпрограммы и пакеты. PL / SQL является расширением методологии SQL-кода Oracle. Подпрограмма PL / SQL представляет собой блок PL / SQL по имени / SQL, который можно вызвать по нескольким параметрам. Пакеты PL / SQL относятся к логическим связанным типам PL / SQL, переменных и подпрограмм.

Другие типы объектов также хранятся в базе данных и могут быть созданы и обработаны в операторах SQL, которые не входят в схему. К ним относятся пользовательские базы данных, задачи, контексты и элементы каталога.

Некоторые объекты в схеме хранят данные в структурах логического хранилища, называемые сегментами. Например, неопределенная таблица кучи или индекса создает сегмент. Другие объекты схемы, например представления и последовательности состоят только из метаданных. В этом случае описываются только объекты меры с сегментами.

База данных Oracle логически хранит объект схемы в пространственной таблице. Связь между схемами и табличными пространствами отсутствует. Вкладки личной области могут содержать объекты из разных схем, а объекты для схемы могут быть вставлены в разные табличные пространства. Данные каждого элемента физически содержатся в одном или нескольких файлах.

На рисунке 2.5 показана возможная конфигурация табличных и индексных сегментов, табличных пространств и файлов данных. Сегмент данных для одной таблицы включает в себя два файла данных, которые являются частью одного и того же табличного пространства. Сегмент не может охватывать более одного табличного пространства.

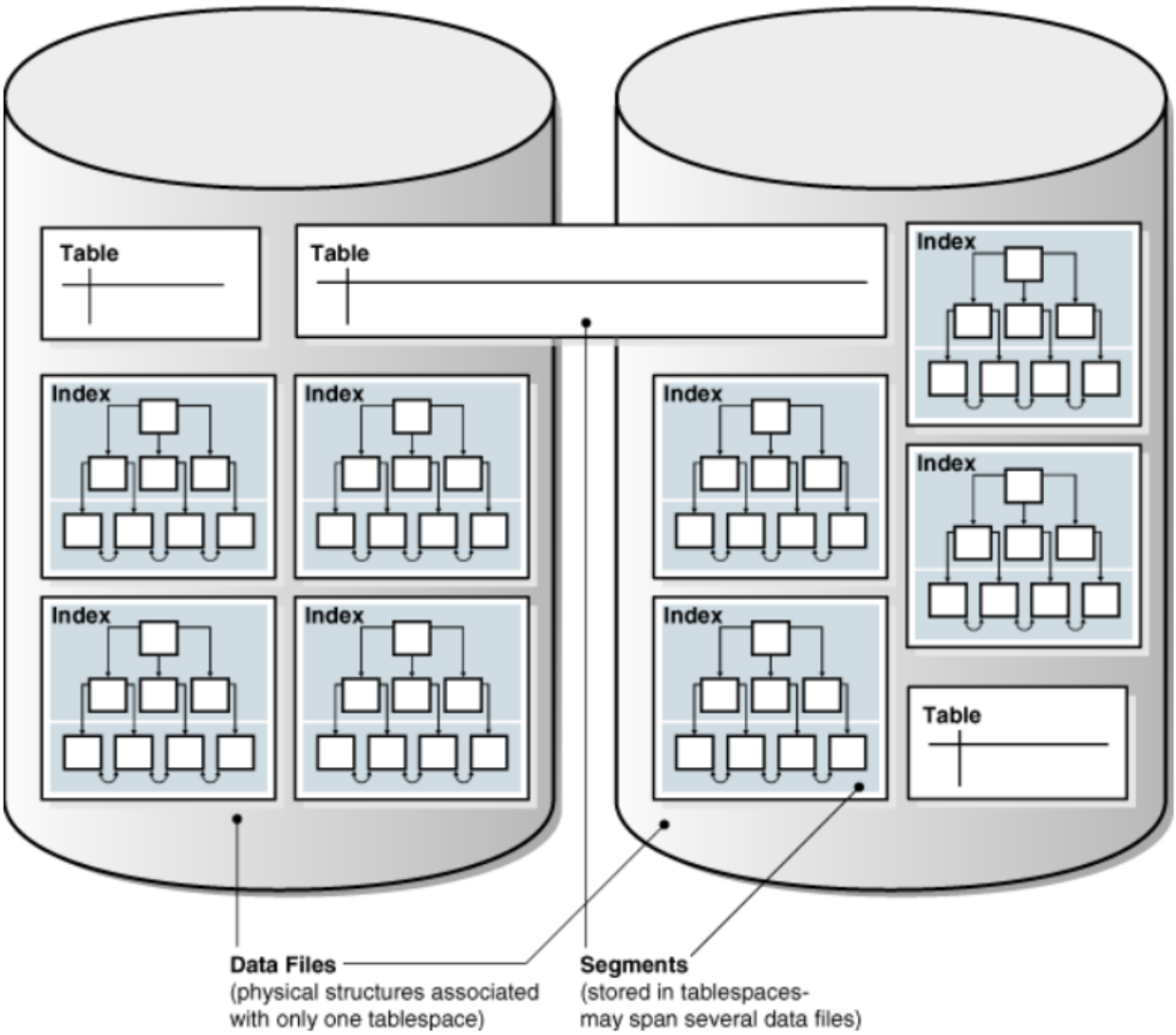


Рисунок 2.5. – Конфигурация табличных и индексных сегментов

Некоторая мера объектов относится к другим вещам и создает зависимости от объекта предложения. Например, представление содержит запрос, который определяет таблицы или другие представления, тогда как подпрограмма PL / SQL вызывает другие подпрограммы.

База данных Oracle обеспечивает автоматизированный механизм, гарантирующий, что зависимые объекты всегда обновляются соответствующими объектами. Когда создается зависимый объект, база данных отслеживает зависимости между надежным объектом и его ссылочными объектами. Если ссылочные позиции меняются, зависимый объект помечен как недопустимый. Например, если пользователь отбрасывает таблицу, удаленный табличный вид невозможен.

Недействительный объект должен быть перекомпилирован на основе нового значения объекта, на который ссылается, используя надежные объекты. Перекомпиляция происходит автоматически при обращении к чему-то, что ненадлежащим образом зависит от объекта.

В качестве иллюстрации того, как объекты схемы могут создавать зависимости, показан рисунок 2.6, где создается таблица table1, а затем процедура, которая запрашивает эту таблицу:

```
CREATE TABLE table1 ( col1 INTEGER, col2 INTEGER );

CREATE OR REPLACE PROCEDURE proc1
AS
BEGIN
 FOR x IN ( SELECT col1, col2 FROM table1 )
 LOOP
   -- process data
   NULL;
 END LOOP;
END;
/
```

Рисунок 2.6. – Создание таблицы table1

Следующий запрос процедуры proc1 показывает, что он действителен:

```
SQL> SELECT OBJECT_NAME, STATUS FROM USER_OBJECTS WHERE OBJECT_NAME = 'PROC1';

OBJECT_NAME STATUS
----------- -------
PROC1       VALID
```

Рисунок 2.7. – Запрос процедуры proc1

После добавления col3 столбца в table1 процедура все еще действительна, поскольку процедура не имеет зависимостей от этого столбца:

```
SQL> ALTER TABLE table1 ADD col3 NUMBER;

Table altered.

SQL> SELECT OBJECT_NAME, STATUS FROM USER_OBJECTS WHERE OBJECT_NAME = 'PROC1';

OBJECT_NAME STATUS
----------- -------
PROC1       VALID
```

Рисунок 2.8. – Добавление столбца col3 и проверка процедуры

Однако изменение типа данных col1 столбца, от которого зависит процедура proc1, приводит к недействительности процедуры:

```
SQL> ALTER TABLE table1 MODIFY col1 VARCHAR2(20);

Table altered.

SQL> SELECT OBJECT_NAME, STATUS FROM USER_OBJECTS WHERE OBJECT_NAME = 'PROC1';

OBJECT_NAME STATUS
----------- -------
PROC1       | INVALID
```

Рисунок 2.9. – Изменение столбца col1 и проверка процедуры

Запуск или повторная компиляция процедуры делает ее действительной снова, как показано в следующем примере:

```
SQL> EXECUTE proc1

PL/SQL procedure successfully completed.

SQL> SELECT OBJECT_NAME, STATUS FROM USER_OBJECTS WHERE OBJECT_NAME = 'PROC1';

OBJECT_NAME STATUS
----------- -------
PROC1       | VALID
```

Рисунок 2.10. – Повторная компиляция

Все базы данных Oracle содержат административные учетные записи. Управления счетом является привилегированным, и предназначены только для администраторов база данных, уполномоченных на выполнение таких задач, как запуск и остановку базы данных, управление хранением и хранение данных, создание и обслуживание базы данных пользователей и т.д.

Учетная запись администратора SYS создается автоматически при создании базы данных. Эта учетная запись может выполнять все административные функции базы данных. Схема SYS хранит основные таблицы и данные для словаря данных. Эти основные таблицы и представления необходимы для работы с базой данных Oracle. Таблицы схем SYS управляются исключительно базой данных и не должны быть изменены пользователем.

Запись SYSTEM автоматически создается при создании базы данных. Схема SYSTEM содержит дополнительные таблицы и представления, которые показывают административную информацию, а также внутренние таблицы и представления, используемые различными параметрами и инструментами Oracle Database. Схема SYSTEM не используется для хранения таблиц, которые интересуются пользователями, не являющимися администраторами.

Таблица является основным элементом данных организации в базе данных Oracle. В таблице описывается объект стоимости, в котором записывается информация. Например, сотрудник может быть юридическим лицом.

Таблицы базы данных Oracle делятся на следующие основные категории:

– реляционные таблицы. Реляционные таблицы имеют простые диапазоны и являются наиболее распространенными типами таблиц;

– таблица элементов. Столбцы соответствуют характеристикам самого высокого уровня типа элемента.

Таблица является постоянной или временной. В повторяющихся таблицах данные хранятся в разных сеансах. Временная таблица хранится так же, как постоянная таблица, но данные существуют только для продолжительности транзакции или сеанса. Временные таблицы полезны для приложений, в которых вы хотите временно сохранить набор результатов, возможно потому, что результат был создан путем выпуска нескольких операций.

Таблица содержит имя таблицы и столбцов. Столбец идентифицирует функцию объекта, описанного в таблице. Например, столбец employees_id в таблице employees определяет атрибут идентификатора сотрудника для сотрудника.

В общем случае при создании таблицы вы указываете имя столбца, один тип данных и одну ширину для каждого столбца. Например, типы данных employees_id - NUMBER (7) указывают, что этот столбец содержит только числовые данные шириной до 7 цифр. Ширина может быть определена как DATE с типом данных.

Таблица может содержать виртуальный столбец без пробела, в отличие от не виртуальных столбцов. База данных принимает значения в виртуальном столбце, если требуется, вычисляя диапазон пользовательских выражений или функций. Например, виртуальная сумма дохода может быть функцией зарплаты и rows_pct.

После создания таблицы вы можете вставлять, запрашивать, удалять и обновлять строки с помощью SQL. Строка представляет собой коллекцию информации о столбцах, которая соответствует записи таблицы. Например, столбец таблицы сотрудников показывает характеристики конкретного сотрудника.

Каждая строка имеет тип данных, связанных с определенным форматом хранения, барьерами и допустимым набором значений. Тип данных связывает фиксированный диапазон функций со значением. Эти атрибуты приносят Oracle Database обрабатывать значения одного типа данных, отличные от значений других типов данных. Например, вы можете умножить NUMBER тип данных, но не тип данных RAW.

Когда вы создаете таблицу, вы должны указать тип данных для каждого столбца. Любое значение, введенное позже в столбце, наследует тип данных столбца.

База данных Oracle предоставляет множество встроенных типов данных. Обычно используемые типы данных относятся к следующим категориям:

– типы символов данных;
– числовые типы данных;
– типы данных datetime;
– типы данных Rowid;
– форматирование моделей и типов данных.

Другие важные категории встроенных типов включают необработанные большие объекты (LOB) и коллекции. PL / SQL имеет типы данных для констант и переменных, содержащих BOOLEAN, ссылочные типы, составные типы (записи) и пользовательские типы.

Символьные типы данных хранят символьные данные (буквенно-цифровые) в строках. Наиболее часто используемым символьным типом является VARCHAR2, который является наиболее эффективным вариантом для хранения символьных данных.

Значения байтов соответствуют схеме кодирования символов, обычно называемой набором символов или кода страницы. Набор символов базы данных задается при создании базы данных. Примерами наборов символов являются 7-разрядные ASCII, EBCDIC и Unicode UTF-8.

Семантика длины символьных типов данных может быть измерена байтами или символами. Байт-семантика рассматривает строки как последовательность байтов. Это значение по умолчанию для типов символьных данных. Символьная семантика рассматривает строки как последовательность символов. Символ технически является кодовой точкой набора символов базы данных.

Таблица кластеров представляет собой группу таблиц с общими столбцами и данными, хранящимися в тех же блоках. Когда таблицы сгруппированы, блок данных может содержать столбцы из нескольких таблиц. Блок может хранить строки, например, вы можете сохранять строки из таблицы Employees и Departments.

Ключом кластера является один или несколько столбцов. Например, таблицы сотрудников и отделов таблицы разделяют столбец department_id. Вы определяете ключ кластера при создании кластера таблиц каждую созданную таблицу, добавленную в кластер.

Значение ключа кластера - это значение столбцов кластера для определенной строки. Все данные, содержащие один и тот же объем ключа кластера, например, Department_id = 20, физически сохранены вместе. Каждое значение ключа кластера сохраняется только один раз в кластере и в индексе кластера, независимо от того, сколько строк в разных таблицах содержит значение.

Аналогичным образом, предположим, что менеджер по персоналу имеет два книжных шкафа: одну из папок сотрудников, а другую - папки отделов. Пользователи часто запрашивают папки для всех сотрудников в определенном отделе. Чтобы облегчить поиск, менеджер приносит все ящики в книжную полку. Она распределяет коробки в соответствии с идентификационным отде-

лом. Таким образом, все папки находятся в отделе 20 для сотрудников и отдела 20 в ящике. Папки для сотрудников отдела 100 и папки для отдела 100 находятся в другой коробке и так далее.

Вы можете просматривать таблицы кластеров, когда они наиболее востребованы (но не изменены), а записи из таблиц часто запрашиваются или агрегируются. Поскольку кластеры хранилища хранят связанные столбцы разных таблиц в обоих блоках данных, правильно используемые кластеры таблиц предлагают следующие преимущества:

– диск ввода / вывода сводится к объединению сгруппированных таблиц;

– время доступа улучшилось для объединения сгруппированных таблиц;

– чтобы сохранить связанный индекс таблицы и данных, требуется меньше памяти, потому что значение ключа кластера не сохраняется снова для каждой строки.

Как правило, таблицы кластеризации не подходят в следующих ситуациях:

– таблицы часто обновляются;

– таблицы часто требуют полноэкранного сканирования;

– столбцы должны быть обрезаны.

Индексированный кластер - это таблица, кластер, которого использует индекс для поиска данных. Индекс кластера - это индекс B-дерева в ключе кластера. Индекс кластера должен быть создан до того, как строки могут быть вставлены в сгруппированные таблицы.

Предположим, что вы создали кластер employees_departments_cluster с ключом кластера department_id, как показано на рисунке 2.11. Поскольку HASHKEYS предложение не указано, этот кластер является индексированным кластером. После этого вы создаете индекс, названный idx_emp_dept_cluster этим ключом кластера.

```
CREATE CLUSTER employees_departments_cluster
   (department_id NUMBER(4))
SIZE 512;

CREATE INDEX idx_emp_dept_cluster ON CLUSTER employees_departments_cluster;
```

Рисунок 2.11. – Создание кластера

Затем вы создаете employees и departments таблицы в кластере, указав department_id столбец как ключ кластера, следующим образом (скобки отмечают место, где идет спецификация столбца):

```
CREATE TABLE employees ( ... )
   CLUSTER employees_departments_cluster (department_id);

CREATE TABLE departments ( ... )
   CLUSTER employees_departments_cluster (department_id);
```

Рисунок 2.12. – Создание кластеров в таблицах employees и departments

Наконец, вы добавляете строки в таблицы employees и departments таблицы. База данных физически хранит все строки для каждого отдела из таблиц employees и departments таблиц в тех же блоках данных. База данных хранит строки в куче и находит их с индексом.

На рисунке 2.13 показан employees_departments_cluster кластер таблиц, содержащий employees и departments. База данных хранит строки для сотрудников в отделе 20 вместе, отдел 110 вместе и т. д. Если таблицы не кластеризованы, база данных не гарантирует, что связанные строки хранятся вместе.

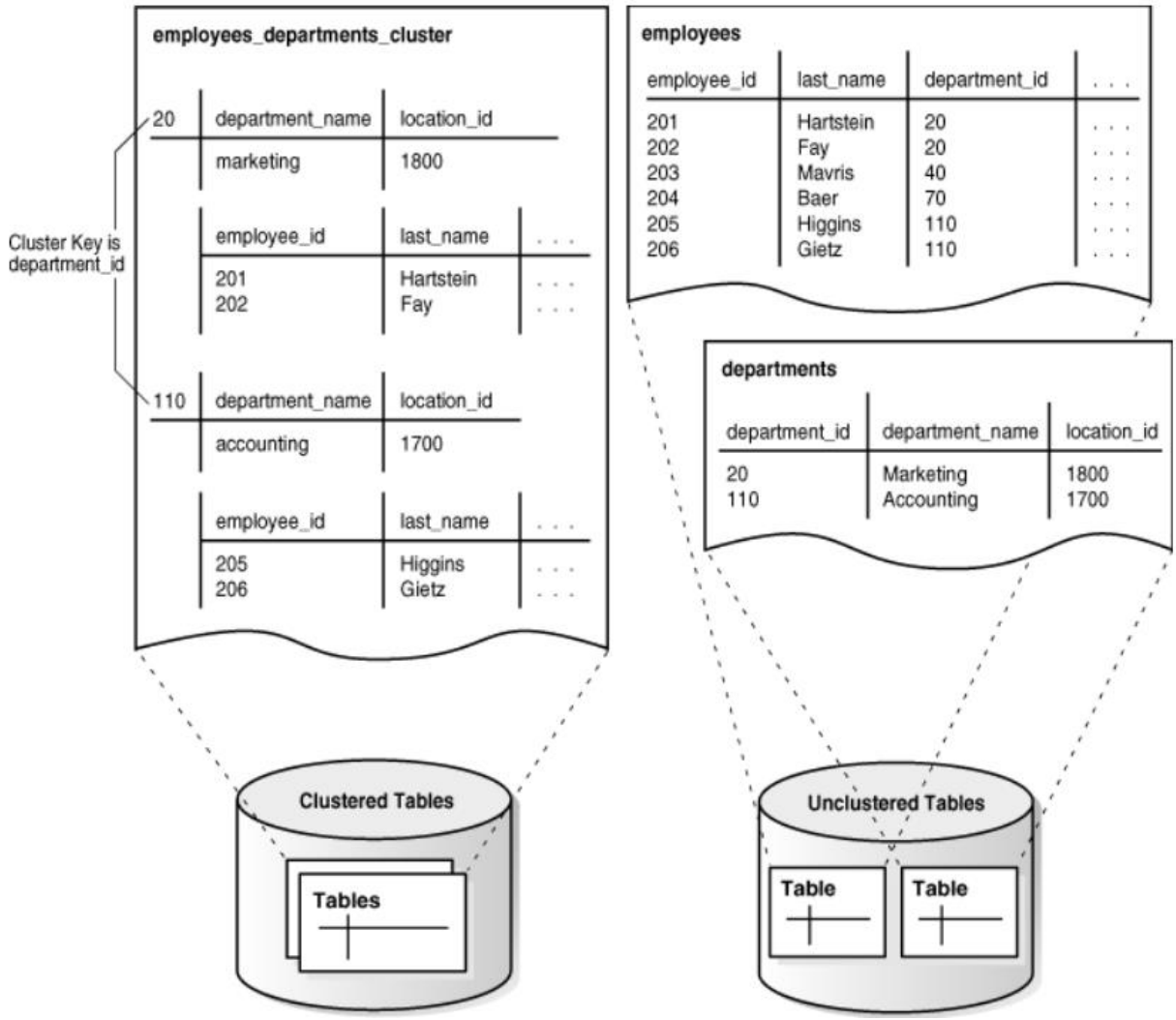


Рисунок 2.13. – Кластер таблиц employees и departments

Хеш-кластер, например, индексированный кластер, отличный от ключа индекса, заменяется хеш-функцией. В хеш-кластере данные являются индексом.

В индексированном или неиндексированном кластере Oracle Database находит таблицы, используя примитивы, хранящиеся в отдельном индексе. Чтобы найти или сохранить строку в индексированной или кластерной таблице, база данных должна иметь как минимум два ввода-вывода:

– один или несколько операций ввода-вывода для поиска или сохранения значений ключа в индексе;

– другой ввод-вывод для чтения или записи строки на столбце или таблице.

Чтобы найти или сохранить строку в хеш-кластере, Oracle Database реализует хеш-функцию в значении ключа кластера строки. Полученное хеш-значение соответствует блоку данных в чтении кластера или записи базы данных от имени совместно используемого оператора.

Хеширование - это дополнительный способ хранения данных таблицы для повышения производительности поиска данных.

Кластерный ключ, как и ключ индексированного кластера, представляет собой столбец или составной ключ, разделяющий таблицы кластеров. Значения ключа хэша являются допустимыми или возможными значениями, которые вводятся в столбцах кластера ключей. Например, если ключ кластера является идентификатором отдела, значения хэш-ключа могут быть 10, 20, 30 и т. д.

База данных Oracle использует хеш-функцию, которая использует бесконечное количество хэш-значений в качестве входных данных и генерирует их с конечным числом разделов. Каждый раздел имеет уникальный числовой идентификатор, называемый хэш-значением. Каждое хеш-значение связано с адресом блока базы данных для блока, в котором хранятся строки, соответствующие значению хэш-ключа (деления 10, 20, 30 и т. д.).

Чтобы создать хэш-кластер, используйте тот же оператор CREATE CLUSTER, что и для индекса кластера, и добавьте хэш-ключ. Количество значений разрушения для кластера зависит от хэш-ключа.

2.2.7 Сравнение с другим СУБД

Каждый администратор баз данных ставит вопрос о том, какая разница между MySQL и Oracle. Оба являются популярными РСУБД. Но функции MySQL и Oracle разные. База данных MySQL и Oracle - это две основные базы данных, которые широко используются во всем мире. Большинство баз данных работают одинаково, но есть некоторые отличия здесь и там.

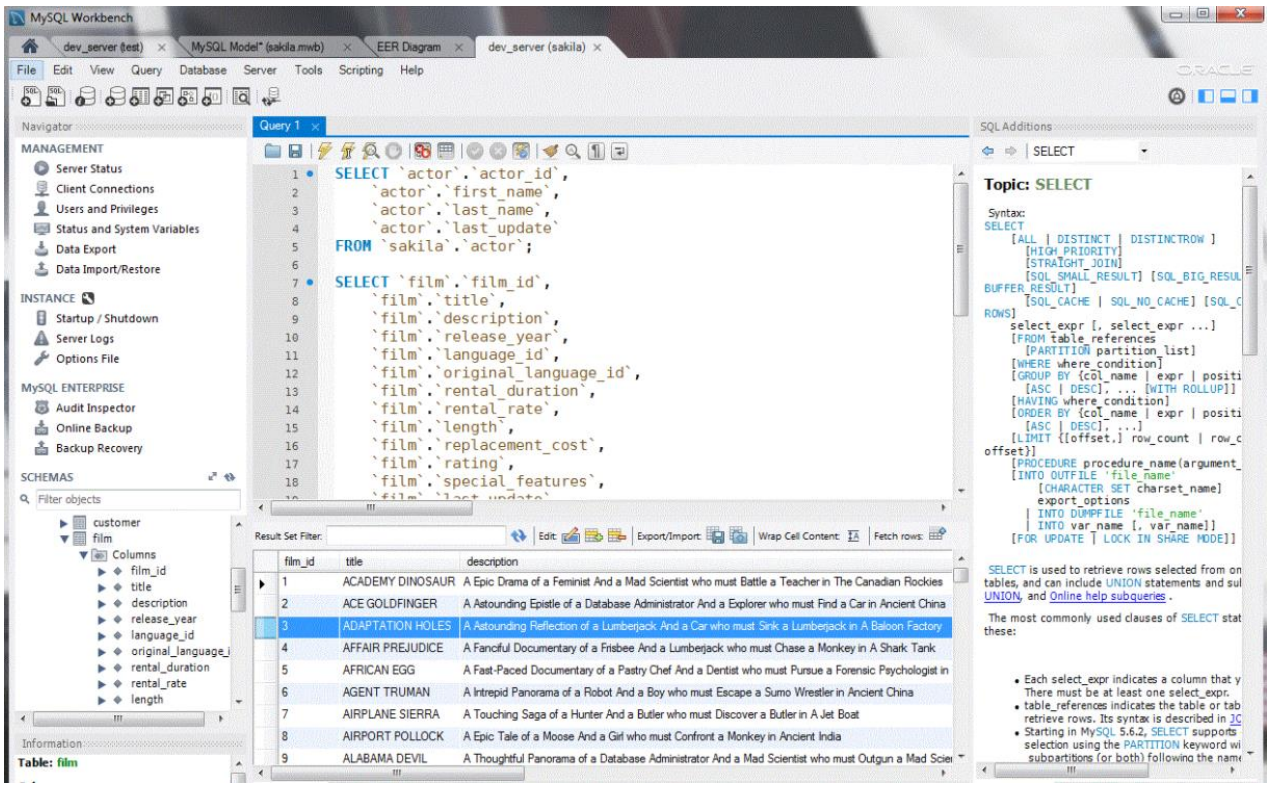


Рисунок 2.14. – Интерфейс БД MySQL

Ниже приведены основные отличия баз данных MySQL и Oracle:

– MySQL использует три параметра для аутентификации пользователя, а именно имя пользователя, пароль и местоположение, Oracle использует так много функций безопасности, как имя пользователя, пароль, профили, локальная аутентификация, внешняя аутентификация, усовершенствования безопасности и т. д;

– синтаксис Oracle SQL существенно отличается от MySQL. Oracle обеспечивает большую гибкость для программирования языка PL/SQL. SQL*Plus инструмент Oracle предлагает больше команд, чем MySQL для генерации вывода отчета и определения переменной;

– в сравнении с Oracle, MySQL не имеет табличного пространства, управления ролями, синонимов и пакетов, а также автоматического управления хранилищем;

– Oracle не зависит от регистра ко всем именам объектов, в то время как некоторые имена объектов MySQL, такие как базы данных и таблицы, чувствительны к регистру (в зависимости от операционной системы);

– база данных Oracle поддерживает несколько языков программирования, которые должны быть записаны, скомпилированы и выполнены изнутри базы данных. Кроме того, для передачи данных используют XML. С другой стороны, MySQL не поддерживает другие языки, выполняемые внутри системы, и он также не поддерживает XML;

– существуют некоторые отличия от типов символов, поддерживаемых в двух базах данных. MySQL имеет CHAR и VARCHAR для типов символов, максимальная длина которых может составлять 65 535 байт (CHAR может быть не более 255 байтов и VARCHAR 65,535 байт). Oracle, с другой стороны, поддерживает четыре типа символов: CHAR, NCHAR, VARCHAR2 и NVARCHAR2. Все четыре типа символов должны иметь минимум 1 байт. CHAR и NCHAR могут быть длиной 2000 байт, а максимальный предел для NVARCHAR2 и VARCHAR2 - 4000 байт;

– база данных MySQL не поддерживает такую функцию, как Audit Vault на своем сервере. Oracle, с другой стороны, поддерживает несколько расширений и программ на своем сервере базы данных, например, Active Data Guard, Audit vault, Partitioning и Data Mining и т. д;

– Oracle и MySQL обрабатывают временные таблицы по-разному. В MySQL временные таблицы видны только для текущего сеанса пользователя, и как только сессия заканчивается, эти таблицы автоматически отбрасываются. Однако в Oracle эти таблицы должны быть явно опущены и видны для всех сеансов;

– Oracle предлагает самую популярную утилиту резервного копирования под названием Recovery Manager (RMAN). Используя RMAN, мы можем автоматизировать нашу базу данных планирования резервного копирования и восстановления, используя очень мало командных или хранимых скриптов. У MySQL есть утилиты резервного копирования mysqldump и mysqlhotcopy. В MySQL нет такой утилиты, как RMAN.

– MySQL более популярен среди веб-сайтов и интеграции PHP. Oracle пользуется большой популярностью в крупных базах данных, таких как Banking, ERP, Insurance, финансовые компании.

– в части администрирования базы данных Oracle DBA больше зарабатывает, чем MySQL DBA. Существует много возможностей для Oracle DBA по сравнению с MySQL.

– сертификация Oracle содержит 3 или более экзаменов для трека DBA и трека разработчика. Сертификация MySQL содержит 1 или 2 экзамена для трека DBA и трека разработчика.

Таблица 2.1 – Сравнение БД Oracle, PostgreSQL и SQL Server

База данных	Oracle	PostgreSQL	SQL Server
Интерфейс	GUI, SQL		GUI, SQL
Поддержка языков	C, C#, C++, Java, Ruby, Objective C и т.д.	.NET, C, C++, Delphi, Java, Perl, PHP, Python, Tcl	Java, Ruby, Python, VB, .NET и PHP
ОС	Windows, Linux, Solaris, HP-UX, ОС X, AIX, z / OS, Solaris	FreeBSD, HP-UX, Linux, NetBSD, OpenBSD, Solaris, Unix, Windows, ОС X	Windows, Linux, ОС X, FreeBSD, Solaris

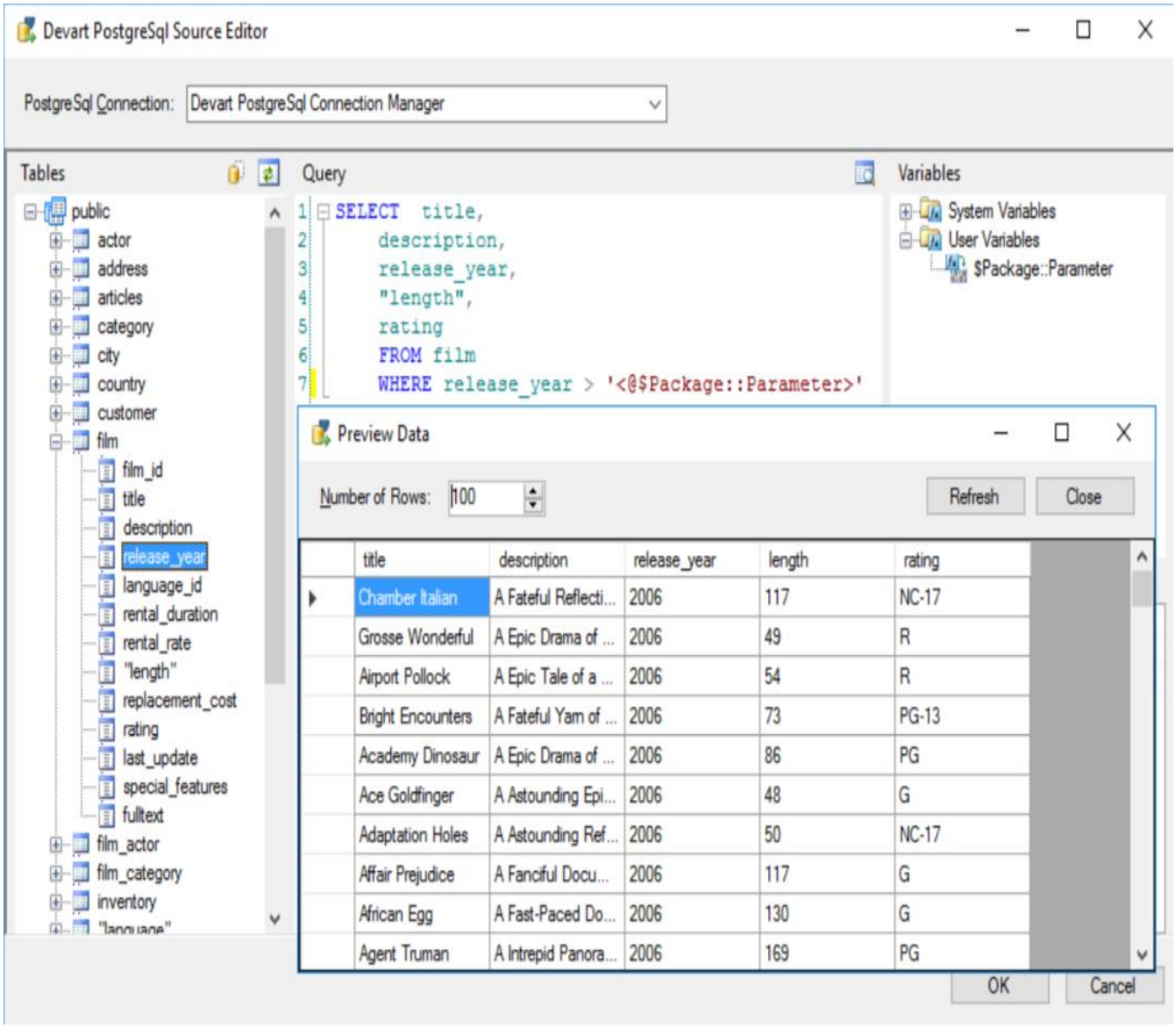


Рисунок 2.15. – Интерфейс БД PostgreSQL

2.3 Visual Studio

Visual Studio - это полный набор средств разработки для создания веб-приложений ASP.NET, веб-служб XML, настольных приложений и мобильных приложений. Visual Basic, Visual C # и Visual C ++ используют одну и ту же интегрированную среду разработки (IDE), которая предоставляет инструменты для совместного использования и позволяет создавать смешанные языковые решения. Кроме того, эти языки используют функциональность .NET Framework, которая обеспечивает доступ к ключевым технологиям, которые упрощают разработку веб-приложений ASP и веб-служб XML.

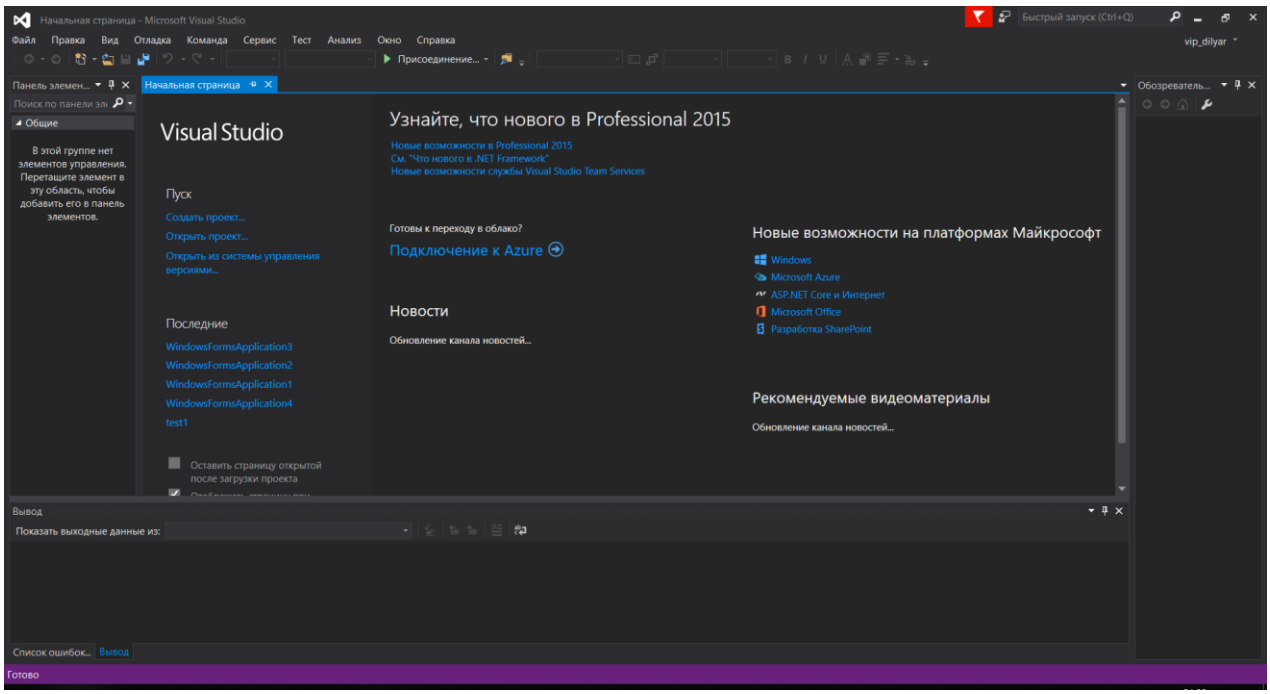


Рисунок 2.16. – Главная страница Visual studio

2.3.1 Windows Forms

Windows Forms - это интеллектуальная клиентская технология для .NET Framework, набор управляемых библиотек, которые упрощают выполнение общих задач приложений, таких как чтение и запись в файловых систем. Если вы используете среду разработки, такую как Visual Studio, вы можете создавать приложения для смарт-клиентов Windows Forms для отображения информации, запроса ввода от пользователей и обмена данными на удаленном компьютере по сетевому адресу.

В Windows Forms форма представляет собой визуальную интерфейс, на которой отображается информация пользователю. Как правило, вы создаете приложения Windows Forms, добавляя формы управления и генерируя ответы на действия пользователя. Как щелчки мыши или нажатия клавиш. Контроллер представляет собой элемент отдельного пользовательского интерфейса (UI), который отображает данные или принимает входные данные.

Когда пользователь запускает что-то в форме или контроле, действие генерирует событие. Приложение реагирует на эти события в коде и обрабатывает события, когда это происходит.

Windows Forms включает в себя множество элементов управления, которые вы можете добавить в форму: управляемые текстовые поля, кнопки, выпадающие переключатели окна и даже увидеть веб-страницы, если существующие элементы управления не отвечают требованиям. Windows Forms также поддерживает создание пользовательских элементов управления с помощью класса UserControl.

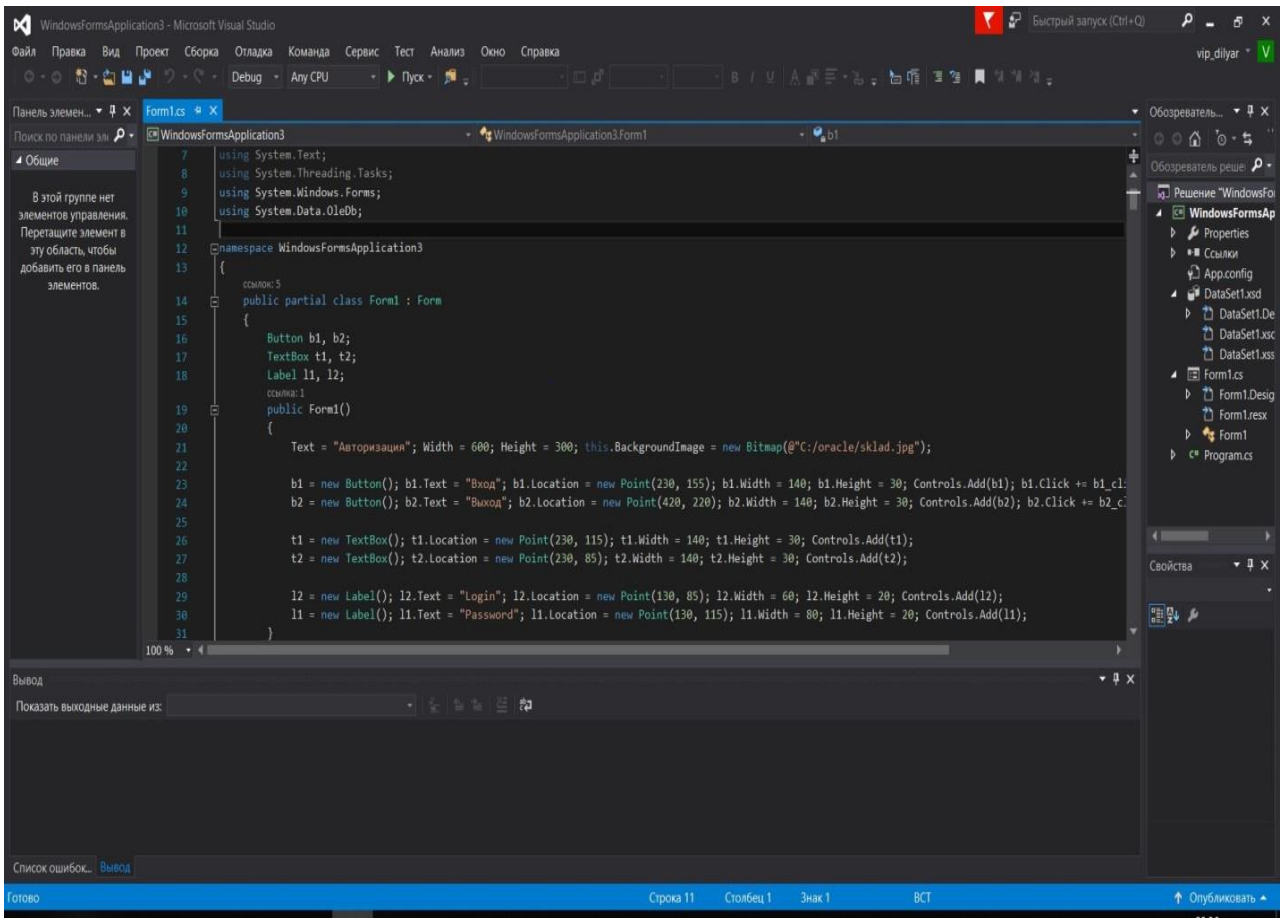


Рисунок 2.17. – Создание графического интерфейса в Windows Forms

У Windows Forms есть богатые элементы пользовательского интерфейса, которые эмулируют функции высокопроизводительных приложений, таких как Microsoft Office. Используя элемент управления и MenuStrip ToolStrip, вы можете создавать панели инструментов и меню, содержащие текст и изображения, отображающие подменю и другие элементы управления, такие как, например, текстовые поля и комбинированные поля.

В Visual Studio Windows Forms Designer вы можете легко создавать приложения Windows Forms. Просто выберите элементы управления курсором, и вы можете добавить их в форму. Дизайнер предоставляет такие инструменты, как линии сетки и привязные линии, чтобы исключить выравнивание элементов управления. И, используя Visual Studio из командной строки или компиляции, вы можете использовать элементы FlowLayoutPanel и TableLayoutPanel SplitContainer для создания расширенных форм компоновки за меньшее время.

Для создания собственных пользовательских элементов интерфейса пространство имен System.Drawing содержит большой выбор классов для визуализации линий, кругов и других фигур непосредственно в форме.

Многие приложения должны отображать данные из базы данных, XML-файла, веб-службы XML или другого источника данных. Windows Forms предоставляет гибкий элемент управления элемента управления, который

является элементом управления DataGridView, который отображает такие данные в формате строк и столбцов записей Tradi Zion, так что каждая часть данных занимает свою собственную ячейку. Если вы используете DataGridView, вы можете настроить внешний вид отдельных ячеек, чтобы блокировать любые столбцы и столбцы на панели, а также отображать сложные элементы управления как внутри ячеек, так и других.

2.3.2 Язык программирования C#

Наименование «C sharp» вдохновлено музыкальной партией, где резкое указывает, что запись должна быть выше в поле. Он похож на имя C++, где «++» указывает, что переменная должна быть увеличена на 1. Острый символ также является лигированием четырех символов «+» («два из двух» «сетка»), что еще больше указывает на то, что язык является плюсом для C++.

Из-за технических ограничений дисплея и того факта, что на раскладках клавиатуры shinstve нет четких символьных номеров символов, резкий символ выбирается в письменных названиях языковых программ для аппроксимации, Это соглашение доступно в спецификации языка ECMA-334 C#. Однако, если это удобно, Microsoft будет использовать значок предполагаемой музыки.

```
{
    Form3 pf;
    Button b;
    Label l1, l2, l3, l4, l5;
    TextBox t1, t2, t3, t4, t5;
    OleDbCommand cmd;
    OleDbConnection con;
    ссылка: 1
    public Form10(Form3 f)
    {
        pf = f;
        Text = "Форма для добавления";
        Width = 220;
        Height = 380;
        con = new OleDbConnection("Provider=MSDAORA;Data Source=orcl;Persist Security Info=True;Password=whouse;User ID=whouse");

        b = new Button();
        b.Text = "Добавить";
        b.Location = new Point(40, 290);
        b.Width = 80;
        b.Height = 30;
        Controls.Add(b);
        b.Click += b_click;

        t1 = new TextBox(); t1.Location = new Point(40, 40); t1.Width = 140; t1.Height = 30; Controls.Add(t1);
        t2 = new TextBox(); t2.Location = new Point(40, 90); t2.Width = 140; t2.Height = 30; Controls.Add(t2);
        t3 = new TextBox(); t3.Location = new Point(40, 140); t3.Width = 140; t3.Height = 30; Controls.Add(t3);
        t4 = new TextBox(); t4.Location = new Point(40, 190); t4.Width = 140; t4.Height = 30; Controls.Add(t4);
        t5 = new TextBox(); t5.Location = new Point(40, 240); t5.Width = 140; t5.Height = 30; Controls.Add(t5);
```

Рисунок 2.18. – Листинг графического интерфейса на языке C#

Суффикс «sharp» используется несколькими другими .NET-языками (разработанными .NET-языками из Microsoft, которые являются производными от Java 1.1), вариантами существующих языков, включая J#, A# (от Ada) и языка функционального программирования F#. Реализация исходного Эйфеля для .NET называется Eiffel#, так как имя было с тех пор, и теперь оно поддерживает весь язык Эйфеля. Используется суффикс (для

оболочки Cocoa) для библиотек, таких как gtk# (.NET-обертка для GTK+ и других библиотек GNOME) и Cocoa#.

Согласно C# Design - язык программирования, который наилучшим образом отражает базовую общую языковую инфраструктуру (CLI). Оно имеет свои встроенные типы, чтобы соответствовать типам значений, реализованных как часть CLI. Однако язык подробностей не вписывает требования для кода и для компилятора, то есть он не показывает, что C должен сосредоточиться на # компиляторе в общем языке времени выполнения или Common Intermediate Language (CIL) или создать один другой конкретный формат. Теоретически компиляторы C# могут генерировать машинный код, например, традиционный компилятор C++ или Fortran. C # полностью объявляет переменные с переменным ключевого слова, которое набирается и одновременно набирает массивы с новым ключевым словом, за которым следует инициализатор коллекции.

C # поддерживает строгий тип данных Boolean, Bool. Выражая, которые принимает такие условия, как while и when выражения, такие как оператор клея Umset, такой как тип Boolean. В то время как C ++ также является логическим, он может свободно преобразовывать в integer и к ним, а выражения, такие как (a) Требуется только конвертировать в bool или Allow int pointer. C # запрещает это «целочисленное значение правильного или неправильного» подхода на том основании, что программист, вынуждающий их использовать выражения, которые возвращают ровно bool, может предотвратить некоторые ошибки программирования, такие как (a = b) (используйте Order вместо равенства «=» - «==», которые, хотя ошибки на C или C ++ не будут, все равно будет компилятором).

C # безопаснее по сравнению с C ++. Единственными неявными преобразованиями, которые считаются по умолчанию, являются безопасные состояния, такие как расширение целых чисел. Это делается во время компиляции во время JIT, а в некоторых случаях во время выполнения. Non-Prois не указывает на преобразование между булевыми и целыми числами между элементами передачи и целыми числами (отличными от буквального 0, которые могут быть неявно преобразованы в любой тип передачи). Преобразование каждого пользователя должно быть четко или четко обозначено, в отличие от строителей и трансформаторов C ++, которые являются высоко стандартизованными.

C # имеет четкую поддержку ковариации и противопоказаний в родовых типах, в отличие от C ++, что позволяет уровень противопоказания поддержки только семантикой, возвращающейся к виртуальным методам Тион.

Элементы доставки размещаются в отдельной рамке.

Язык C # не разрешает международные переменные или функции. Все методы и члены должны быть объявлены внутри классов. Статические члены могут заменять различные классы pub переменными и глобальными функциями. Локальные переменные не могут теневые переменные вложенного блока, в отличие от C и C ++.

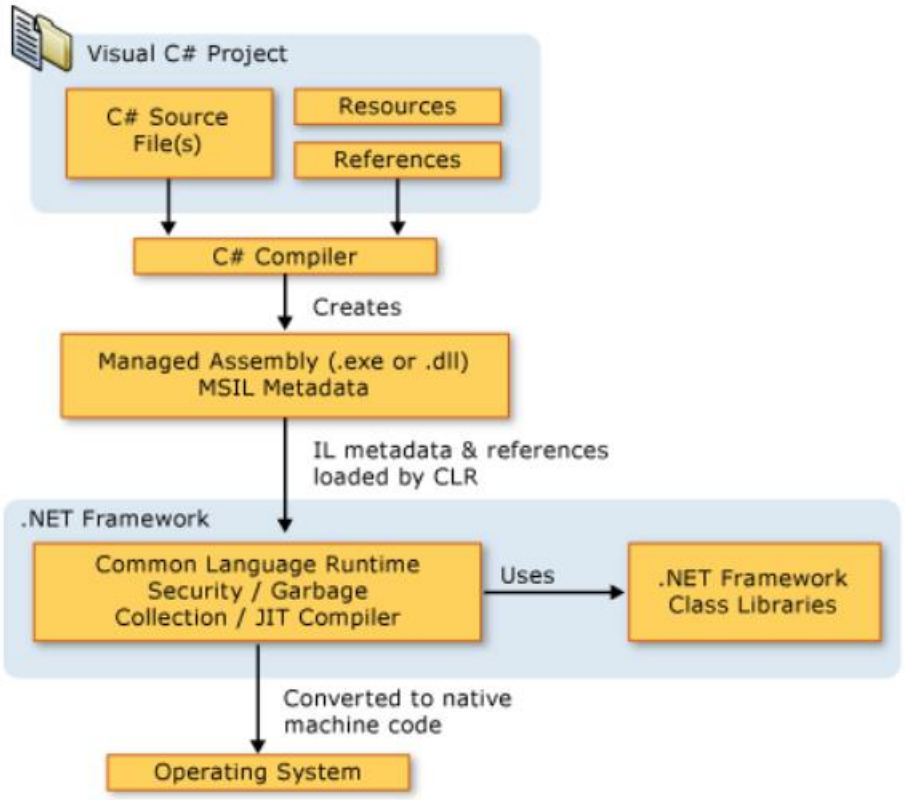


Рисунок 2.19. – Отношения времени компиляции исходного кода C #

C # предлагает Java-подобный метод через атрибут [MethodImpl (MethodImplOptions.Synchronized)] и поддерживает эксклюзивные замки с ключевым замком.

C # представляет атрибуты, такие как синтаксической атрибут для устаревшего рисунка, полученного с парами процедур доступа, и устанавливаются одним классом атрибута. Любые избыточные подписи, которые обеспечивают соблюдение Gether / Seter, не требуется, и, следовательно, могут быть доступны с помощью синтаксиса атрибутов вместо метода вызова, имеющего дробную часть.

C # пространства имен обеспечивает тот же уровень Java изоляции код пакета или C ++ пространство имен с очень похожими правилами и функциями для пакета.

В C #, ссылки адреса памяти могут быть использованы только в единицах, которые специально помечены как небезопасные и небезопасных программ, которые имеют соответствующие разрешения для запуска кода. В основном доступ к объекту через ссылку, чтобы обеспечить объекты, которые всегда относятся к «живому» объекту или имеют четко определенное нулевое значение. Невозможно получить доказательство «мертвого» объекта (собранные отходы) или непредусмотренного блок памяти.

Небезопасный указатель может указывать на то время, к значению массива, строки или блок памяти, который связан со стеком.

2.4 SQL Developer

Свободный графический интерфейс пользователя Oracle SQL Developer позволяет пользователям и администраторам баз данных выполнять свои задачи в базе данных с меньшим количеством кликов и нажатий клавиш. Инструмент увеличения производительности, где основная цель SQL Developer - экономить время конечных пользователей и максимизировать отдачу от инвестиций в технологию Oracle Database Stack.

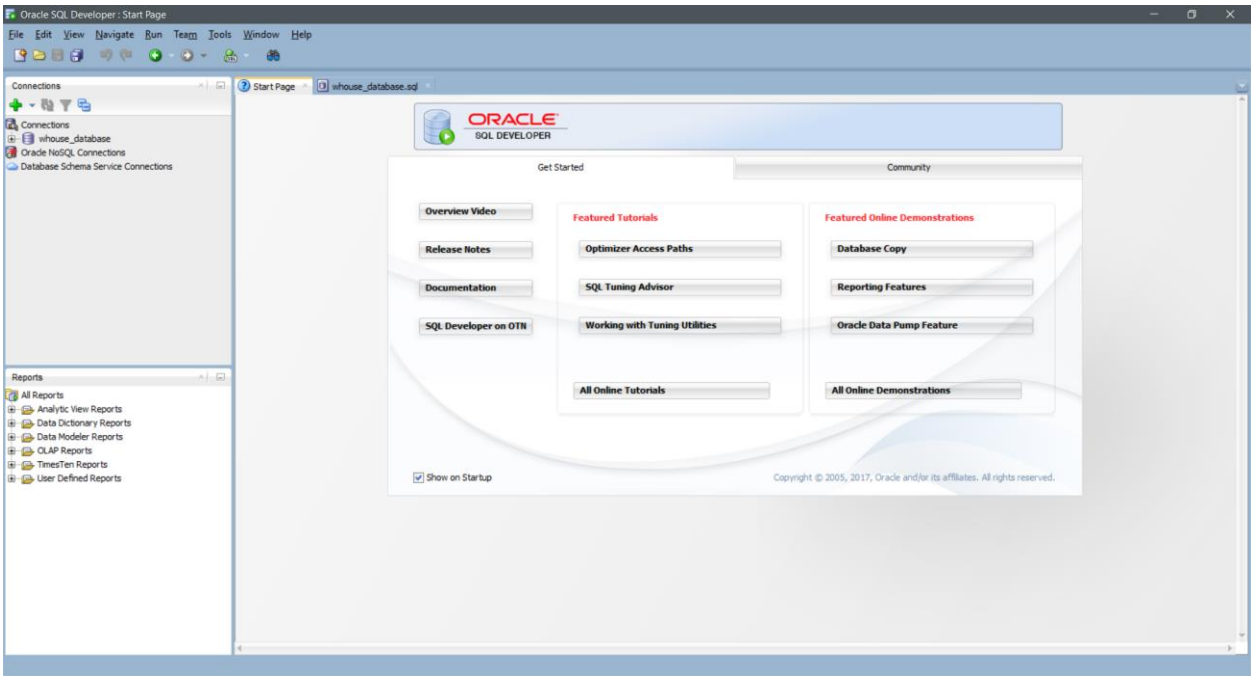


Рисунок 2.20. – Главный экран SQL Developer

SQL Developer поддерживает Oracle Database 10g, 11g и 12c и будет работать в любой операционной системе, поддерживающей Java.

3 Практическая часть

3.1 ER-диаграмма базы данных производственного склада

Модель ER-диаграммы иллюстрирует смежные объекты, которые заинтересованы в определенной области знаний. Модель ER состоит из типов объектов, которые классифицируют объекты, представляющие интерес, и определяют отношения, которые могут возникать между экземплярами этого объекта.

Модель компонента программного обеспечения ER обычно предназначена для представления действия, которое программа должна учитывать при следующем запуске фреймворка для действия и порядка команд. Таким образом, модель ER представляет собой абстрактную диаграмму данных или информацию, которые могут быть реализованы в базе данных, обычно используя реляционный дизайн и построение.

Модель ER также может быть выражена словами, например, здание можно разделить на один или более комнат, но одну комнату можно найти только в доме или здании.

Сущности могут быть идентифицированы не только отношениями, но и дополнительными активами, другими словами, атрибутами, которые содержат идентификаторы, называемые «первичными ключами».

Графики, созданные для представления атрибутов, сущностей и отношений, можно назвать диаграммами сущностей «сущность-связь», а не моделями отношений.

В простом смысле каждая строка таблицы представляет собой экземпляр базы данных типа сущности, и каждое поле в таблице представляет тип атрибута. В реляционной базе данных связь между объектами реализуется путем хранения первичного ключа чего-то в качестве указателя на другой объект. Он создает стандарт для создания моделей диаграмм ER, данных на двух или трех уровнях абстракции. Концептуально-иерархическая иерархия используется в других типах спецификаций и отличается от трехсхемного стратегического программирования и программного обеспечения.

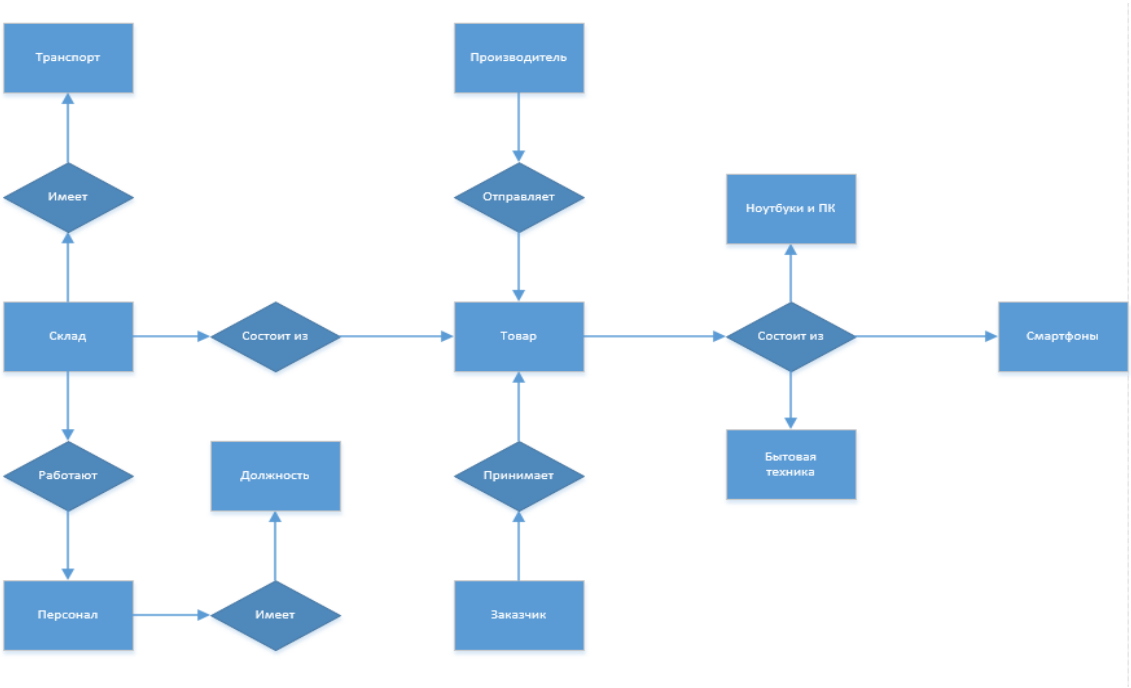


Рисунок 3.1. – ER-диграмма

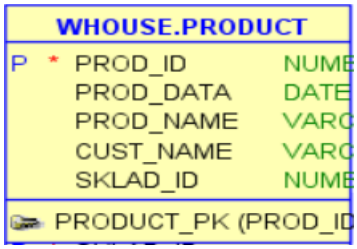
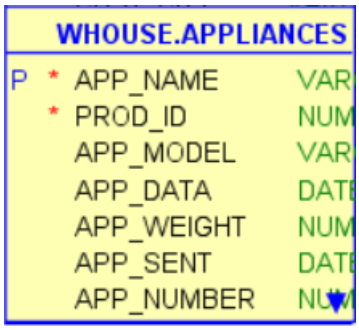
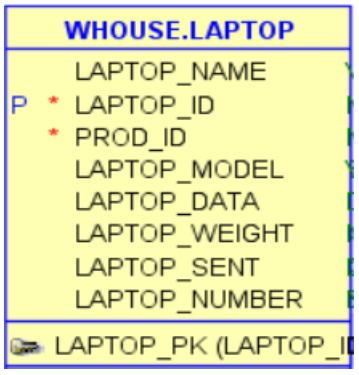
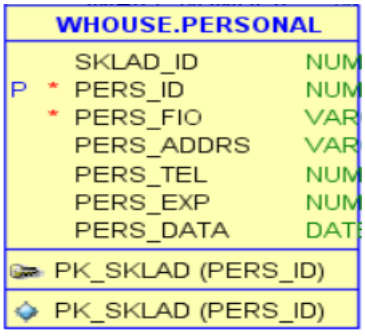
3.2 Назначение всех сущностей проекта

В таблице 3.1 описываются все сущности, присутствующие в таблицах базы данных склада.

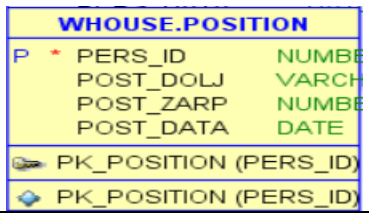
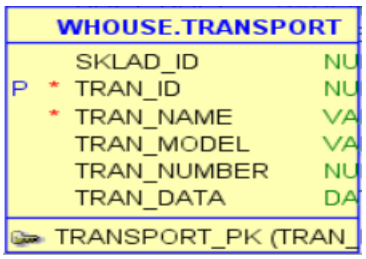
Таблица 3.1 – Реестр всех сущностей

Сущность	Описание
WHOUSE.SKLAD P * SKLAD_ID * SKLAD_NAME * SKLAD_ADDRESS TELEPHONE SKLAD_CITY SKLAD_PK (SKLAD_ID)	Таблица «sklad» содержит: SKLAD_ID – идентификатор склада SKLAD_NAME – название склада SKLAD_ADDRESS – адрес склада TELEPHONE – телефон склада SKLAD_CITY – город в котором находится склад
WHOUSE.PRODUCER P * PROD_NAME CUST_COUNTRY PRODUCER_PK (PROD_	Таблица «producer» содержит: PROD_NAME – название производителя CUST_COUNTRY – страна производитель
WHOUSE.CUSTOMER P * CUST_NAME VARC CUST_ADD VARC CUST_CITY VARC CUST_TEL NUME CUSTOMER_PK (CUST_	Таблица «customer» содержит: CUST_NAME – имя заказчика CUST_ADD – адрес заказчика CUST_CITY – город в который направится товар CUST_TEL – телефон заказчика

Продолжение таблицы 3.1

	Таблица «product» содержит: PROD_ID – идентификатор товара PROD_DATA – дата получения товара PROD_NAME – название производителя CUST_NAME – имя заказчика SKLAD_ID – идентификатор склада
	Таблица «appliances» содержит: APP_NAME – название техники PROD_ID – идентификатор товара APP_MODEL – модель техники APP_DATA – дата производства APP_WEIGHT – вес техники APP_SENT – дата отправки APP_NUMBER – номер техники
	Таблица «smartphone» содержит: SMART_NAME – название смартфона SMART_ID – идентификатор склада PROD_ID – идентификатор товара SMART_MODEL – модель смартфона SMART_DATA – дата производства SMART_WEIGHT – вес смартфона SMART_SENT – дата отправки SMART_NUMBER – номер смартфона
	Таблица «laptop» содержит: LAPTOP_NAME – название ПК LAPTOP_ID – идентификатор ПК PROD_ID – идентификатор товара LAPTOP_MODEL – модель ПК LAPTOP_DATA – дата производства LAPTOP_WEIGHT – вес ПК SMART_SENT – дата отправки SMART_NUMBER – номер ПК
	Таблица «personal» содержит: PERS_ID – идентификатор сотрудника SKLAD_ID – идентификатор склада PERS_FIO – ФИО сотрудника PERS_ADDRS – адрес проживания PERS_TEL – телефон сотрудника PERS_EXP – опыт работы PERS_DATA – дата рождения

Продолжение таблицы 3.1

	Таблица «position» содержит: PERS_ID – идентификатор должности POST_DOLJ – наименование должности POST_ZARP – заработная плата POST_DATA – дата выпуска должности
	Таблица «transport» содержит: SKLAD_ID – идентификатор склада TRAN_ID – идентификатор транспорта TRAN_NAME – название транспорта TRAN_MODEL – модель транспорта TRAN_NUMBER – номер транспорта TRAN_DATA – дата производства

3.3 Описание атрибутов сущностей

Таблица 3.2 – Атрибуты сущности «sklad»

Название поля	Описание атрибута	Тип	Длина	Примечание
ID склада	sklad_id	number	10	Primary key
Название склада	sklad_name	varchar2	30	Not null
Адрес склада	sklad_address	varchar2	70	Not null
Телефон склада	telephone	numeric	10	Null
Город склада	sklad_city	varchar2	20	Null

Таблица 3.3 – Атрибуты сущности «producer»

Название поля	Описание атрибута	Тип	Длина	Примечание
Название производителя	prod_name	varchar2	15	Primary key
Страна производитель	cust_country	varchar2	20	Null

Таблица 3.4 – Атрибуты сущности «customer»

Название поля	Описание атрибута	Тип	Длина	Примечание
ФИО заказчика	Cust_name	Varchar2	15	Primary key
Адрес заказчика	Cust_add	Varchar2	25	Null

Продолжение таблицы 3.4

Город заказчика	Cust_city	Varchar2	20	Null
Телефон заказчика	Cust_tel	Numeric	10	Null

Таблица 3.5 – Атрибуты сущности «product»

Название поля	Описание атрибута	Тип	Длина	Примечание
ID товара	Prod_id	Number	10	Primary key
Дата изготовки товара	Prod_data	Date	-	Null
Название производителя	Prod_name	Varchar2	15	References producer
ФИО заказчика	Cust_name	Varchar2	15	References customer
ID склада	Sklad_id	Number	10	References sklad

Таблица 3.6 – Атрибуты сущности «appliances»

Название поля	Описание атрибута	Тип	Длина	Примечание
Название техники	App_name	Varchar2	20	Primary key
ID товара	Prod_id	Number	10	References product
Модель техники	App_model	Varchar2	20	Null
Дата производства	App_data	Date	-	Null
Вес техники	App_weight	Number	5	Null
Дата отправки	App_sent	Date	-	Null
Номер техники	App_number	Number	10	Null

Таблица 3.7 – Атрибуты сущности «smartphone»

Название поля	Описание атрибута	Тип	Длина	Примечание
Название смартфона/ телефона	Smart_name	Varchar2	20	Null

Продолжение таблицы 3.7

ID смартфона	Smart_id	Number	10	Primary key
ID товара	Prod_id	Number	10	References
Модель смартфона	Smart_model	Varchar2	20	Null
Дата производства	Smart_data	Date	-	Null
Вес смартфона	Smart_weight	Number	5	Null
Дата отправки	Smart_sent	Date	-	Null
Номер смартфона	Smart_number	Number	10	Null

Таблица 3.8 – Атрибуты сущности «laptop»

Название поля	Описание атрибута	Тип	Длина	Примечание
Название ПК	Laptop_name	Varchar2	20	Null
ID ПК	Laptop_id	Number	10	Primary key
ID товара	Prod_id	Number	10	References product
Модель ПК	Laptop_model	Varchar2	20	Null
Дата производства	Laptop_data	Date	-	Null
Вес ПК	Laptop_weight	Number	5	Null
Дата отправки	Laptop_sent	Date	-	Null
Номер ПК	Laptop_number	Number	10	Null

Таблица 3.9 – Атрибуты сущности «personal»

Название поля	Описание атрибута	Тип	Длина	Примечание
ID склада	Sklad_id	Number	10	References sklad
ID сотрудника	Pers_id	Number	10	Not null
ФИО сотрудника	Pers_fio	Varchar2	20	Not null
Адрес сотрудника	Pers_addrs	Varchar2	70	Null
Телефон сотрудника	Pers_tel	Numeric	10	Null
Опыт работы	Pers_exp	Number	10	Null
Дата рождения	Pers_data	Date	-	Null

Таблица 3.10 – Атрибуты сущности «position»

Название поля	Описание атрибута	Тип	Длина	Примечание
ID сотрудника	Pers_id	Number	10	Not null
Должность	Post_dolj	Varchar2	20	Null
Зарплата	Post_zarp	Number	20	Null
Дата выпуска должности	Post_data	Date	-	Null

Таблица 3.11 – Атрибуты сущности «transport»

Название поля	Описание атрибута	Тип	Длина	Примечание
ID склада	Sklad_id	Number	10	References sklad
ID транспорта	Tran_id	Number	10	Primary key
Название транспорта	Tran_name	Varchar2	20	Not null
Модель транспорта	Tran_model	Varchar2	20	Null
Номер транспорта	Tran_number	Number	15	Null
Дата производства	Tran_data	Date	-	Null

3.4 Работа с базой данных

3.4.1 Создание и заполнение базы данных в SQL Developer

Настроив связь между SQL Developer и Oracle 10g, создаем подключение под названием whouse_database.

После создания подключения, создаем таблицы. Для этого во вкладке «whouse_database.sql» пишем код для создания таблиц с указанием первичных и вторичных ключей и связей между ними (рисунок 3.2).

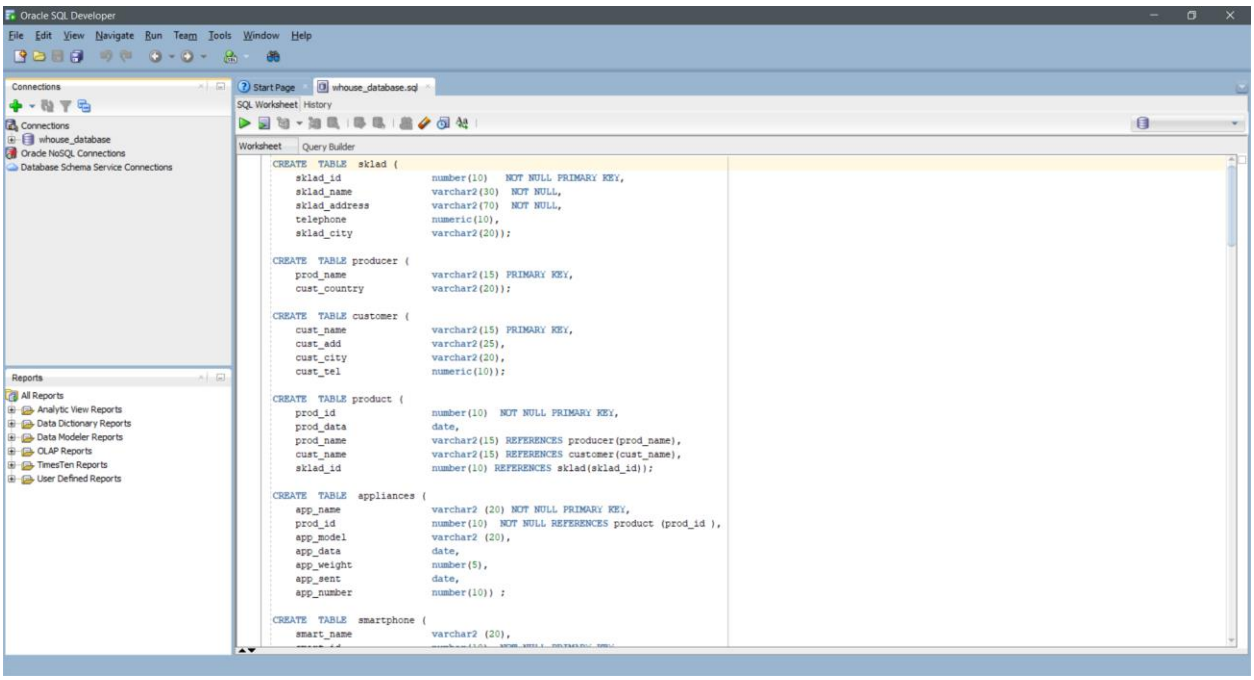


Рисунок 3.2. – Создание таблиц

Созданные таблицы отображаются в левом окне (рисунок 3.3).

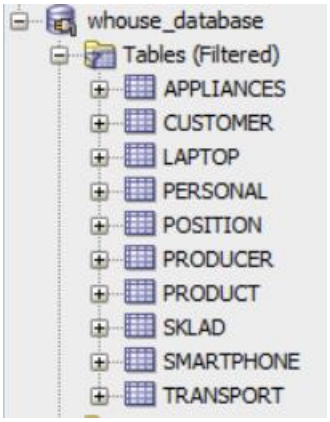


Рисунок 3.3. – Таблицы

Благодаря простоте работы в SQL Developer в файле «whouse_database.sql», помимо создания, можно также заполнить все таблицы (рисунок 3.4).

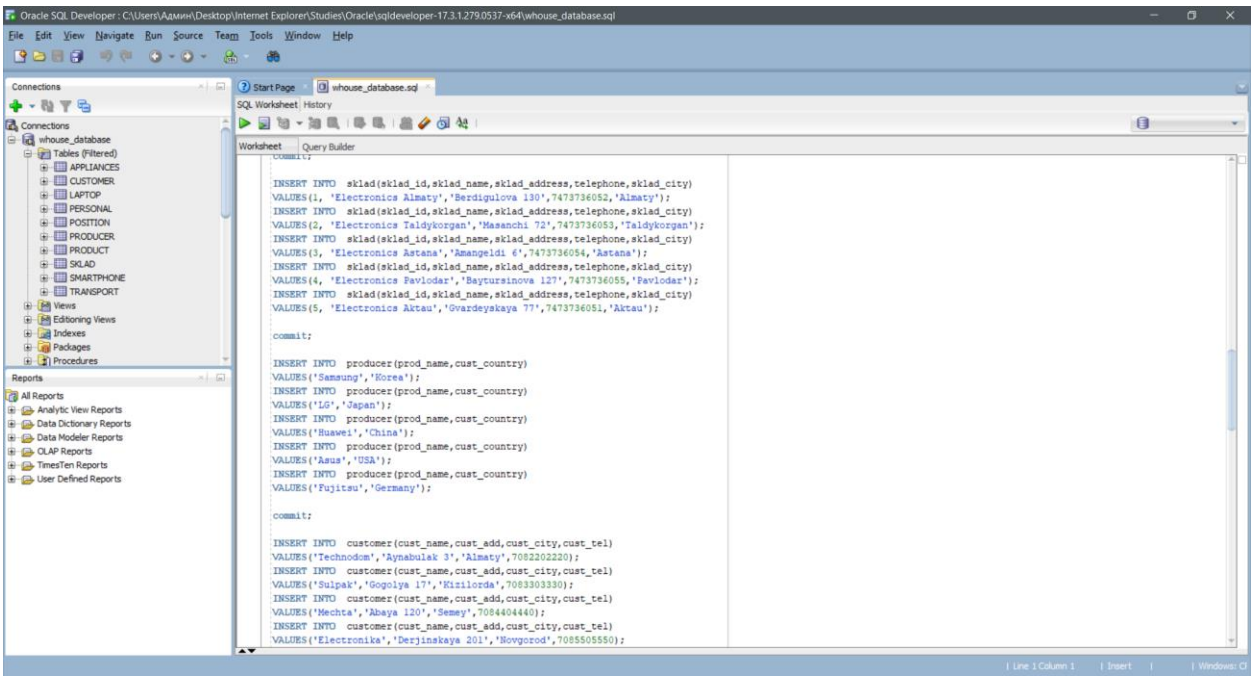


Рисунок 3.4. – Пример заполнения таблиц

3.4.2 Графический интерфейс базы данных в Windows Forms

Данное приложение разработана с учетом интуитивно-понятного интерфейса.

С помощью визуального приложения администратору будет проще вводить информацию. На рисунке 3.5 показана форма авторизации пользователя.

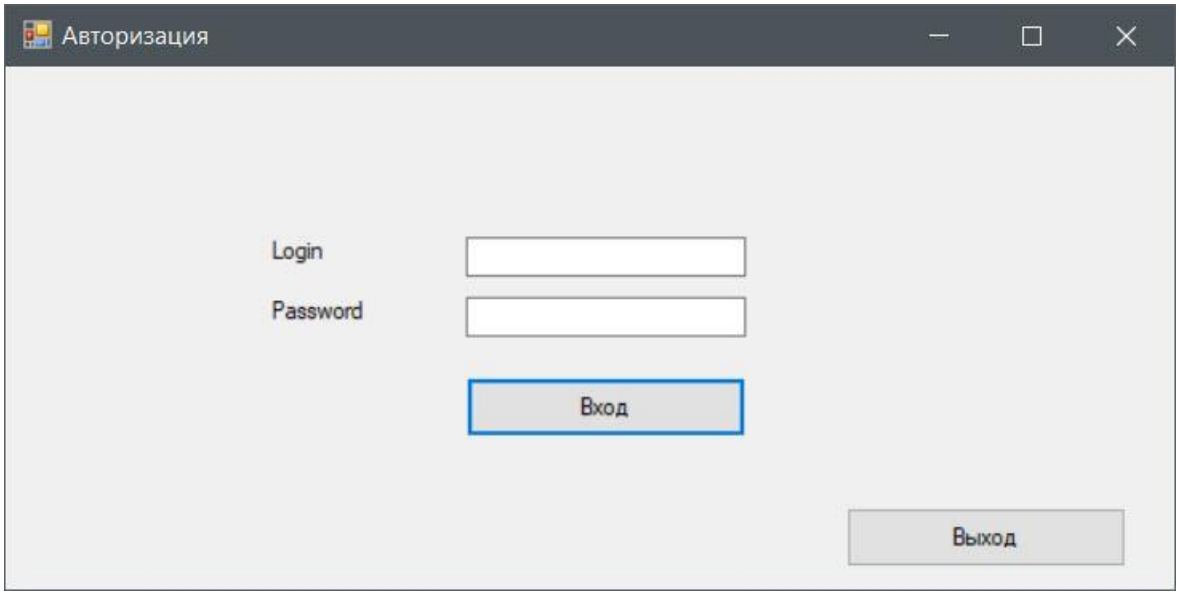


Рисунок 3.5. – Форма авторизации

Интуитивно понятный интерфейс - это пользовательский интерфейс, который должен чувствовать себя комфортно при получении информации, он не должен тратить слишком много времени на его поиск. Пользовательский интерфейс должен использовать только простые формы и мягкие цвета. Это позволяет уменьшить напряжение в глазах пользователя и тщательно изучить, кто использует систему.

Пользователь вводит свои входные данные, т.е. Логин и Пароль. Затем нажимает на кнопку «Вход» или же может отказаться и выйти из программы, нажав на кнопку «Выход».

При неправильном вводе логина и пароля высвечивается окошка, где сообщается, что данные введены неправильно (рисунок 3.6).

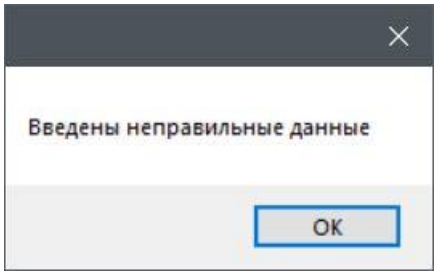


Рисунок 3.6. – Сообщение о неправильном вводе

Если логин и пароль введены правильно, то высвечивается окошко, об удачной авторизации (рисунок 3.7).

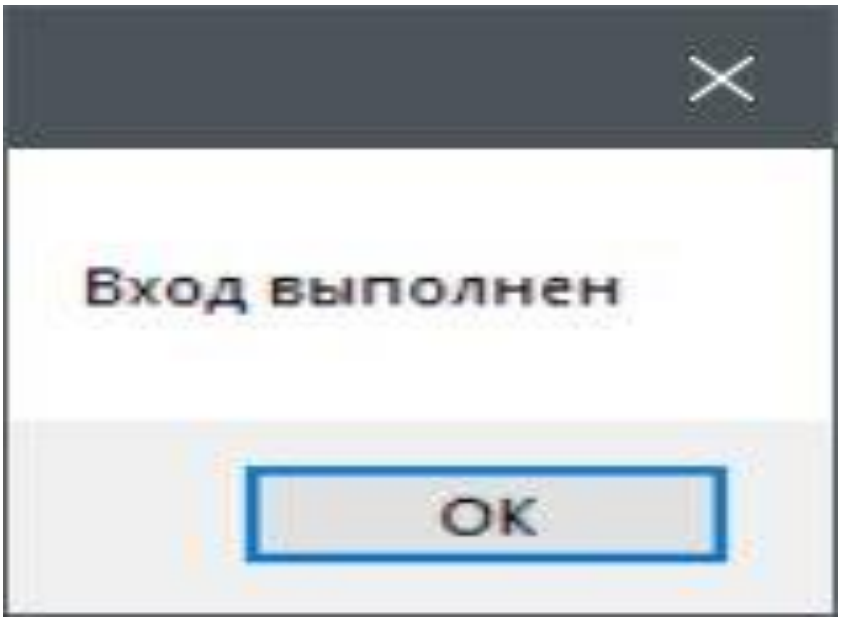


Рисунок 3.7. – Сообщение о входе в систему

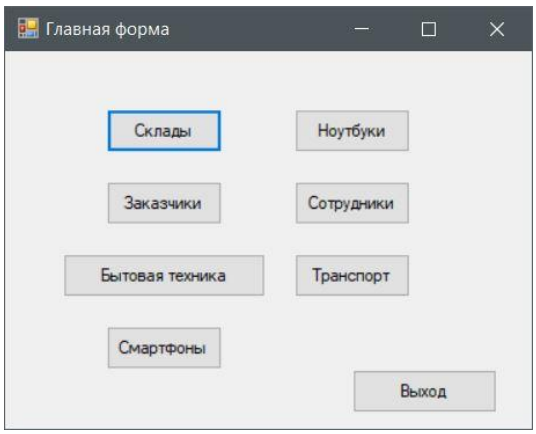


Рисунок 3.8. – Главная форма

В главной форме (рисунок 3.8) имеются расположены кнопки «Склады», «Заказчики», «Бытовая техника», «Смартфоны», «Ноутбуки и ПК», «Сотрудники», «Транспорт» и «Выход». Это позволит администратору с легкостью выбирать таблицы, в которые необходимо внести данные. При нажатии на кнопку администратор может просматривать таблицы базы данных, вносить добавления и удаления данных, а также обновлять таблицу и ввести поиск по конкретному полю.

Форма «Склады» (рисунок 3.9) упрощает возможность отображения данных о производителях. Это играет немалую роль для предприятия, ведь предприятие в будущем будет расширяться, и открывать различные точки в разных городах. Открытие новых точек способствует быстрой доставке, таким образом, список заказчиков будет расширяться.

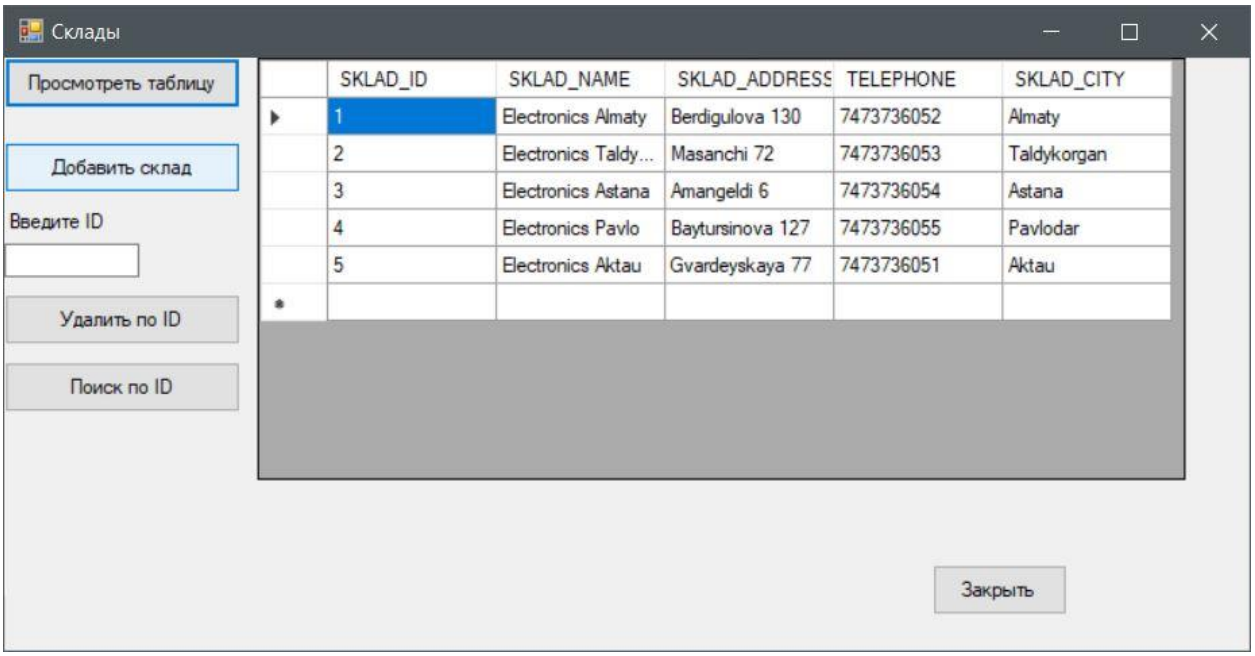


Рисунок 3.9. – Форма «Склады»

В форме расположены кнопки «Просмотреть таблицу», «Добавить склад», «Удалить по ID» и «Поиск по ID», а также текстовое поле для поиска по ID.

При нажатии на кнопку «Просмотреть таблицу» в правом окошке выводятся данные таблицы «sklad». Для добавления данных в таблицу имеется кнопка «Добавить» и при нажатии на нее появляется форма «Форма для добавления» (рисунок 3.10).

В форме «Форма для добавления» вводятся все данные, которые показаны в сущности склада. Данные вводятся в специальные текстовые поля, поверх которых указана название столбца таблицы. Для добавления в форме предусмотрена кнопка «Добавить». Вводимая информация попадает в базу данных, это позволяет администратору добавлять данные, не открывая саму БД Oracle.

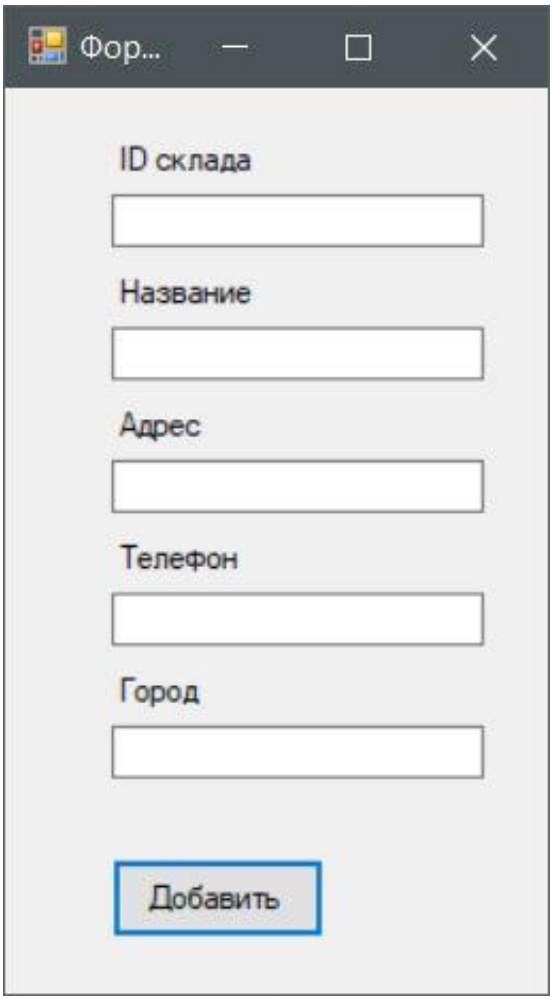


Рисунок 3.10. – Форма для добавления

После нажатия на кнопку «Добавить» высвечивается окошко, подтверждающая о внесении данных в базу (рисунок 3.11).

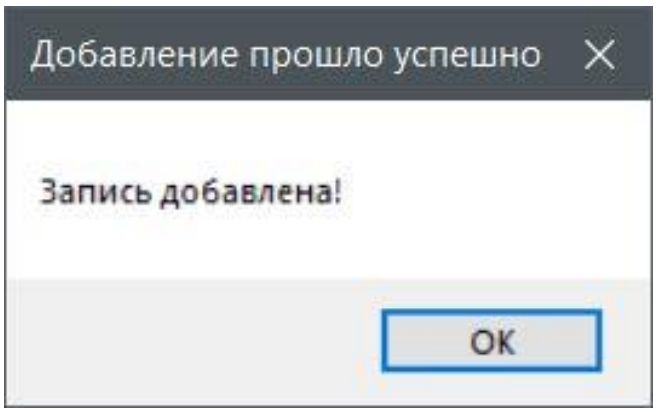


Рисунок 3.11. – Окно подтверждения

Также в форме «Склады» можно ввести поиск по ID. Для этого вводится ID склада в текстовом поле и нажимается кнопка «Поиск по ID» (рисунок 3.12). Эта функция минимизирует поиск по БД и упрощает работу администратора.

ID склада это уникальный идентификатор, который присваивается для каждого склада в разном виде, таким образом, исключается возможность удаления более одного склада.

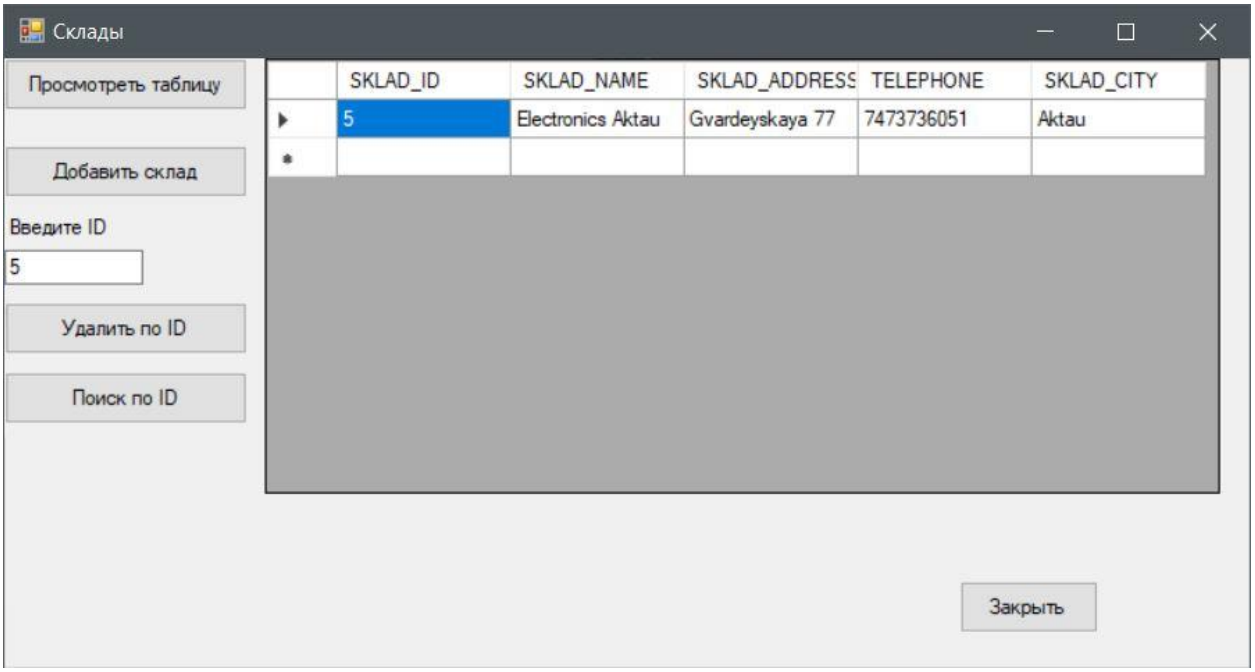


Рисунок 3.12. – Пример выполнения поиска

Кнопка «Удалить по ID» обеспечивает удаление строки по уникальному идентификатору склада.

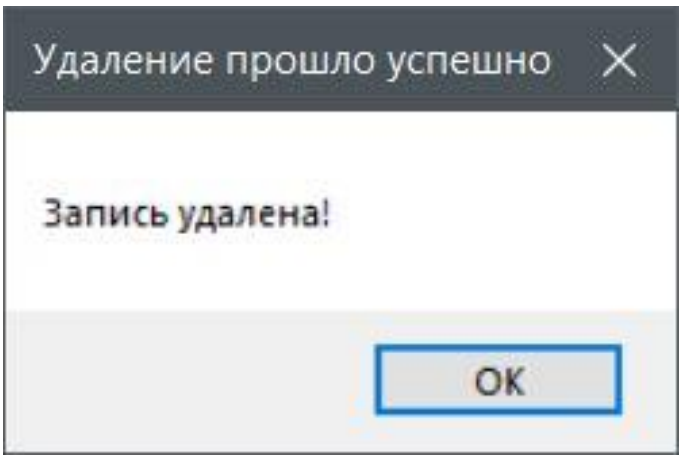


Рисунок 3.13. – Пример удаления записи

4 Безопасность жизнедеятельности

В данной дипломной работе рассматривается разработка базы данных производственного склада. База данных будет написано с помощью СУБД Oracle, также будет разработан графический интерфейс в среде программирования Visual Studio.

База данных будет разрабатываться в офисном помещении «Департамент разработки» и при работе используется компьютерной техника и электронное оборудование. В рассматриваемом помещении работают специалист по БД, оператор и администратор, которые имеют свое рабочее место.

В связи с этим целесообразно создать оптимальные условия для разработки, безопасности и здоровья разработчика.

4.1 Анализ условий труда в офисном помещении «Департамент разработки»

В современную эпоху организации сталкиваются с рядом проблем из-за динамичного характера окружающей среды. Одна из многих задач для бизнеса – обеспечить безопасность своих сотрудников.

Цель данной главы анализировать влияние рабочей среды на безопасность работников. Многие предприятия не понимают важности безопасности рабочей среды для удовлетворенности работой сотрудников и, таким образом, сталкиваются с большими трудностями во время своей работы.

С целью создания нормальных производственной среды сотрудников следует установить кондиционеры. При работе с ПК должны соблюдаться следующие условия:

В холодный период:

– оптимальная температура 22 - 24 °С , допустимая – 18 - 26°С;

– относительная влажность воздуха 40 - 60 %, допустимая 75 %.

В теплый период:

– оптимальная температура 23-25 °С, допустимая 20 - 30°С;

– относительная влажность воздуха 40-60 %, допустимая влажность 55%.

В кабинетах, где расположены персональные компьютеры, особенно важна система пожаротушения. Производственные факторы вызывающие возгорание:

– наиболее высокая температура поверхностей ПК;

– выделение ряда химических веществ в воздух рабочей зоны;

– повышенное значение напряжения в электрической цепи, замыкание;

– повышенный уровень статического электричества;

– повышенный уровень электромагнитных излучений;

– повышенная напряженность электрического поля.

При малейшем замыкании есть шанс возгорания. Для предотвращения очагов возгорания в рабочем месте установлены спринклерные установки.

Также немалую роль играет система эвакуации. Каждый работник предприятия должен ознакомиться с планом эвакуации для предотвращения лишних действий в случае чрезвычайных случаев.

Рассмотрим анализ пожарной эвакуации, связанный с приложениями контроля дыма. При наличии систем наполнения дымом у рабочих должно быть достаточно времени, чтобы покинуть помещение, прежде чем дым опустится на них. Во время пожаров в зданиях лифты почти всегда выводятся из эксплуатации, а вертикальная эвакуация происходит через лестницы. В некоторых ситуациях лифты используются для эвакуации, но так как на складе не предусмотрен лифт, то расчет будет посвящен только вертикальной эвакуации через лестницы.

Описание рабочего кабинета:

– рабочее место расположено на 1 этаже;

– размер комнаты и 4x6x3 (ширина, длина, высота);

– источники света являются по 4 люминесцентные лампы;

– в помещении два окна, которые создают благоприятные условия для зрительного восприятия;

– жалюзи, для защиты от избыточной яркости с окон;

– в рабочем помещении работают 3 человека.

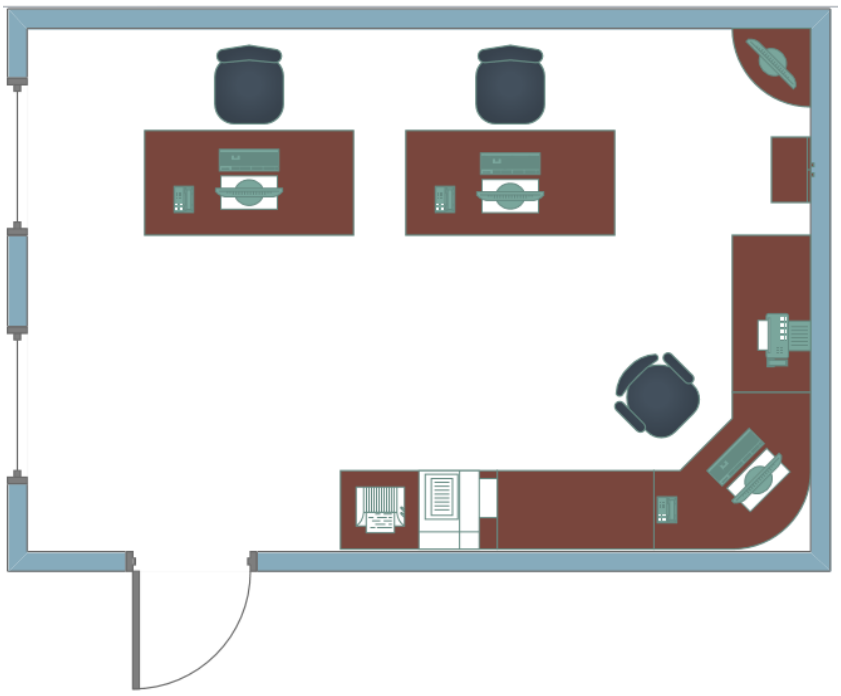


Рисунок 4.1. – План рабочего кабинета

4.2 Расчет систем кондиционирования рабочего помещения сотрудников

Ниже приведен расчет кондиционирования воздуха на рабочих местах сотрудников. Кондиционер обеспечит соответствие рабочей среды всем стандартам.

Количество приточного воздуха $L_{ПР}$, м^3/ч определяем по формуле

$$L_{ПР} = \frac{Q_{ИЗБ}}{c \cdot p_{ПР} \cdot (t_{ВЫТ} - t_{ПР})}, \quad (4.1)$$

где $Q_{ИЗБ}$ - избыточное выделение явной теплоты, кДж/ч;

c - удельная теплоемкость воздуха при постоянном давлении, равная 1кДж/(кг × ˚С);

$p_{ПР}$ - плотность поступающего в помещение воздуха, равная 1,2 кг/м^3;

$t_{ВЫТ}$ - температура удаляемого из помещения за пределы рабочей или обслуживаемой зоны, ˚С;

$t_{ПР}$ - температура приточного воздуха, ˚С;

Температура удаляемого из помещения воздуха $t_{ВЫТ}$, ˚С, определяется по формуле

$$t_{ВЫТ} = t_{РЗ} + \Delta t \cdot (h - z), \quad (4.2)$$

где $t_{РЗ}$ - температура в рабочей зоне, которая не должна превышать допустимую по нормам ($t_{РЗ} \leq t_{ДОП}$), ˚С. Поскольку расчет производится для теплого периода года, то примем $t_{РЗ}$ = 22 ˚С;

Δt - температурный градиент по высоте помещения, ˚С;

h - расстояние от пола до центра вытяжных проемов (кондиционера), м. Внутренняя часть кондиционера расположена на высоте h = 2,7 м;

z- высота рабочей зоны, м.

$t_{ВЫТ} = 22 + 1{,}2 \times (2{,}7 - 3) = 21{,}64$˚С.

Температура приточного воздуха ($t_{ПР}$) при наличии при наличии избытка явной теплоты должна быть на 5-7˚С ниже температуры воздуха в рабочей зоне.

$t_{ПР} = 22 - 6 = 16$˚С.

Величину избыточного выделения явной теплоты $Q_{ИЗБ}$ находят на основании баланса теплоты в помещении по формуле:

$$Q_{ИЗБ} = \sum Q - \sum Q_{УХ}, \quad (4.3)$$

где $\sum Q$ - суммарное количество поступающей в помещение явной теплоты;

$\sum Q_{УХ}$ - суммарное количество уходящей из помещения теплоты (за счет теплопотерь ограждениями, нагрева поступающего в помещение воздуха и т. п.).

Основными источниками перегрева являются ленты, промышленные печи, электроустановки, люди и т. д. В то же время нельзя забывать о потреблении тепла от солнечной радиации.

Вычисляется тепловая мощность искусственного освещения Q_2, определяется, что, выполняя всю потребленную энергию, она в конечном итоге преобразуется в тепло по формуле:

$$Q_2 = 1000 \times N, \quad (4.4)$$

где N - расходуемая мощность светильников, Вт.

$Q_2 = 1000 \times 0{,}25 \times 2 = 500$ Вт.

Тепловыделения от людей Q_3 определяют по формуле:

$$Q_3 = n \times q, \quad (4.5)$$

где n - число работающих;

q - количество тепла, выделяемое одним человеком, Вт.

Таблица 4.1 - Количество тепла, выделяемое одним человеком в зависимости от категории работ и температуры окружающей среды

Категория работ	Тепло, Вт			
	Полное		Явное	
	При 10 ˚С	При 35 ˚С	При 10 ˚С	При 35 ˚С
Легкая	180	145	150	5

$Q_3 = 3 \times 145 = 435$ Вт.

Количество тепла, поступающего в помещение от солнечной радиации $Q_{ОСТ.РАД}$, определяют по формуле:

$$Q_{ОСТ.РАД} = F_{ОСТ} \times q_{ОСТ} \times A_{ОСТ} \quad (4.6)$$

Для покрытий

$$Q_{П.РАД} = F_n \times q_n \times k_n, \quad (4.7)$$

где $F_{ОСТ}$ и Fn - площадь поверхности и покрытия, м^2;

$q_{ОСТ}$ и q_n - теплопоступления через 1 м^2 поверхности остекления и поверхности покрытия, при коэффициенте теплопередачи, равном 1 Вт/(м^2 × ˚С);

$A_{ОСТ}$ - коэффициент остекления;

k_n - коэффициент теплопередачи покрытия, Вт/(м^2 × ˚С).

Значение $q_{ОСТ}$ в зависимости от географической местоположения поверхности и характеристики окон или ламп принимается в пределах 70 - 210, а коэффициента $A_{ОСТ}$ в зависимости от вида остекления и его солнцезащитных функций - в пределах 0,25 - 1,25, средние значения энергии от солнечной радиации через покрытие в зависимости от географической долготы и вида покрытия принимают в пределах 6 - 24.

$F_{ОСТ} = 2 \times 1{,}8 = 3{,}6$ м^2.

Окно в рабочем помещении смотрит на север, поэтому примем значение $q_{ОСТ}$ равным 160 Вт/(м^2 × ˚С). Поскольку остекление образовано светлым стеклом и без стального переплетения, то примем $A_{ОСТ} = 0{,}4$.

$Q_{ОСТ.РАД} = 3{,}6 \times 160 \times 0{,}4 = 230{,}4$ Вт.

Среднее значение теплопоступления для покрытия с учетом географической долготы примем равным $Q_{ОСТ.РАД} = 18$Вт.

Количество потери тепла из рабочего кабинета $Q_{УХ}$, кВт через стены двери, окна насчитывается примерно по следующей формуле:

$$Q_{УХ} = \frac{\lambda \cdot S \cdot (t_{ВЫТ} - t_{ПР})}{\delta}, \quad (4.8)$$

где λ - теплопроводность стен, Вт/(м × ˚С);

S - площадь, м^2;

δ - толщина стен, м.

Стены рабочего места сделаны из тяжелого бетона М510, теплопроводимость которого равна 1,2 Вт/(м × ˚С). Толщина стен $\delta = 0{,}5$м.

$Q_{УХ} = \frac{1{,}2 \cdot 24 \cdot (21{,}64 - 16)}{0{,}5} = 325$ Вт.

Определим сумму общего числа теплоты, что поступает в кабинет

$$\sum Q = Q_2 + Q_3 + Q_{ОСТ.РАД} + Q_{П.РАД} \quad (4.9)$$

$\sum Q = 500 + 435 + 230{,}4 + 18 = 1183{,}4$Вт.

Подставляя значения, вычислим избыточное выделение явной теплоты

$Q_{ИЗБ} = 1183,4 – 325 = 858,4.$

Также определим количество приточного воздуха

$$L_{пр} = \frac{858,4}{1 \cdot 1,2 \cdot (21,64 - 16)} = 126,8 \ \frac{м^3}{ч}.$$

Таким образом, ориентируясь на расчетах в разделе БЖД решил, что необходим кондиционер в рабочем кабинете сотрудников: Panasonic CS/CU-UE12RKD - мультисплит-система настенного типа, японского качества, что имеет в комплекте надежный компрессор и качественными комплектующими для, непосредственно, долгой и бесперебойной работы.

Данный экземпляр владеет саморегулируемыми створками, которые отвечают за полную циркуляцию воздуха в рабочем кабинете.

– мощность при обогреве - 1100 Вт;
– мощность при охлаждении - 1090 Вт;
– расход воздуха - до 762 $м^3$/час;
– уровень шума < 42 Дб;
– габариты внутреннего блока 290х870х214 мм, вес - 9 кг (внутренний блок), 33 кг (наружный блок);
– рассчитан на помещение площадью до 35 $м^2$.

4.3 Расчет систем пожаротушения

Гидравлический расчет выполняется в соответствии с требованиями СП 5.13130.2009 «Установки пожаротушения и сигнализации. Нормы и правила проектирования».

Для защиты производственной среды выбираем ороситель СВО0-РНд0,84-R1/2/Р141.В3-"СВН-К80" с коэффициентом производительности k = 0,84 (по тех.документации на ороситель).

Таблица 4.2 - Группы помещений (производств и технологических процессов) по степени опасности развития пожара в зависимости от их функционального назначения и пожарной нагрузки сгораемых материалов

Группа помещений	Перечень характерных помещений, производств, технологических процессов
1	Помещения книгохранилищ, библиотек, цирков, хранения сгораемых музейных ценностей, фондохранилищ, музеев и выставок, картинных галерей, концертных и киноконцертных залов, ЭВМ, магазинов, зданий управлений, гостиниц, больниц.

Как видно из графика (рисунок 4.2) для интенсивности орошения 0,08 дм^3/м^{2}с подходят три типа оросителя – «СВН-К115», «СВН-К80» и «СВН-К57». Выбирают ороситель, который обеспечивает заданную интенсивность при должном давлении, в нашем случае это «СВН-К80» по паспорту СВО0-РНд0,84-R1/2/Р141.В3 - (диаметр выходного отверстия 10мм., коэффициент производительности К = 0,84).

Таблица 4.3 – Исходные данные для группы помещения 5.

Группа помещений		1
Интенсивность орошения защищаемой площади, л/(с·м^2), не менее	Водой	0,08
	Раствором пенообразователя	-
Расход, л/с, не менее	Воды	10
	Раствора пенообразователя	-
Минимальная площадь спринклерной АУП, м^2, не менее		60
Продолжительность подачи воды, мин, не менее		30
Максимальное расстояние между спринклерными оросителями, м		4

Интенсивность орошения оросителя «СВН-К80» равна 0,08 дм^3/м^{2}с при давлении 0,1 МПа (рисунок 4.2). Тушение огня при возгорании производится с помощью раствора пенообразователя.

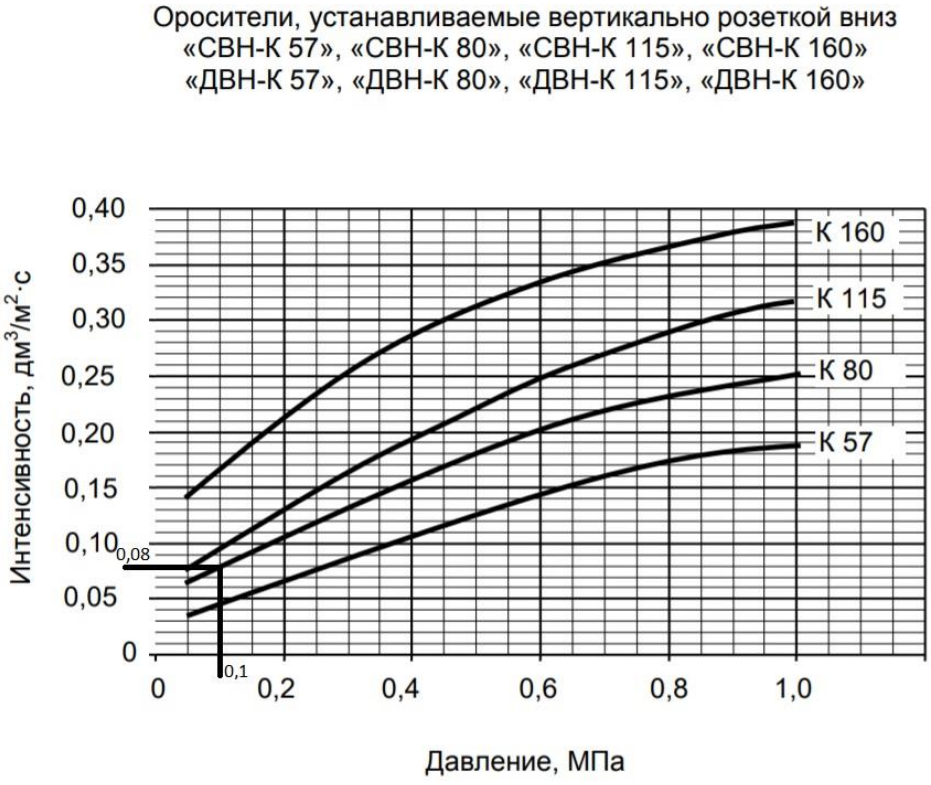


Рисунок 4.2. – Интенсивность орошения

Расчетный расход воды через диктующий ороситель, расположенный в диктующей защищаемой орошаемой площади, определяем по формуле

$$q_1 = 10K\sqrt{P} \quad (4.10)$$

$q_1 = 10 \times 0{,}84\sqrt{0{,}1} = 2{,}52$ л/с.

где q_1 - расход ОТВ через диктующий ороситель, л/с;

К - коэффициент производительности оросителя, принимаемый по технической документации на изделие, л/(с×МПа^{0,5});

Р - давление перед оросителем, МПа.

При произведении гидравлического расчета в учебных целях, предлагается определять количество оросителей для защиты минимальной диктующей площади и их расстановки по формуле:

$$n = \frac{Q_н}{q_1} \quad (4.12)$$

где q_1 — расход ОТВ через диктующий ороситель, л/с;

$Q_н$ — нормативный расход спринклерной АУП согласно таблице 4.3.

$n = \frac{30}{2{,}52} = 12$ шт.

Расположение спринклеров в выбранном минимальном диапазоне диктовки показано на рисунке 4.3. При настройке следует учитывать, что расстояние между спринклерами не должно превышать нормативное расстояние, указанное в таблице 4.3.

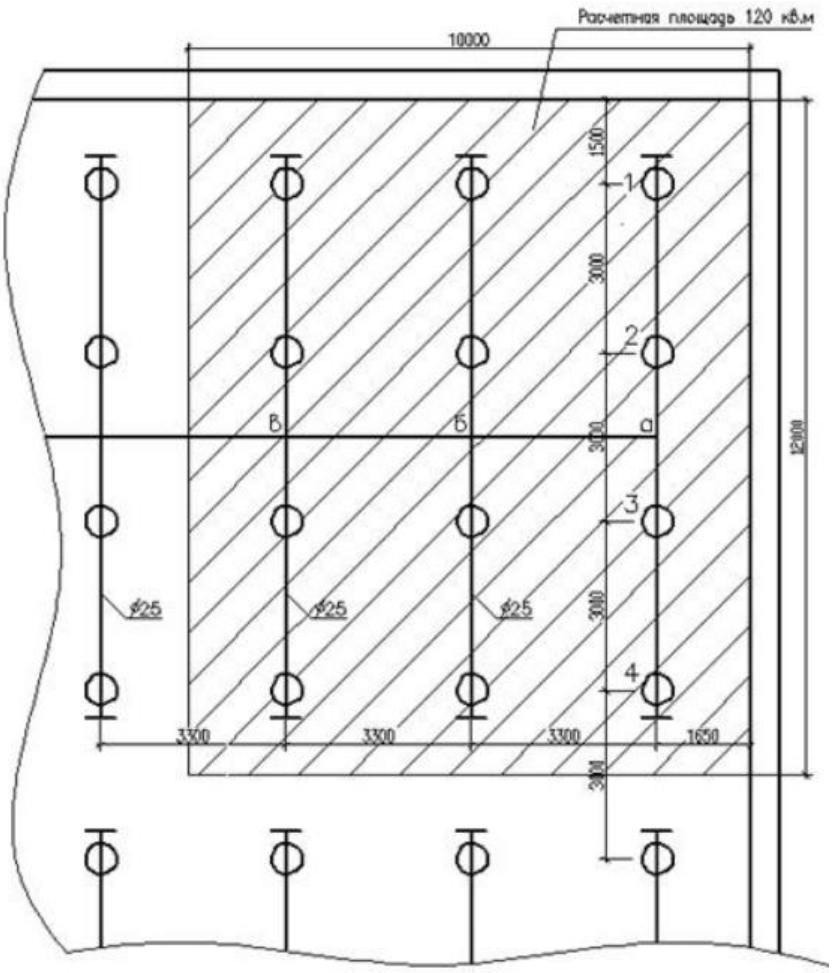


Рисунок 4.3. – Схема размещения оросителей

4.4 Расчет систем эвакуации

4.4.1 Общее время эвакуации

Время эвакуации состоит из времени предварительного движения и времени движения:

$$t_t = t_{pm} + n \times t_m, \qquad (4.13)$$

где t_t = общее время эвакуации (в минутах);

t_{pm} = время предварительного движения (в минутах);

t_m = смоделированное время эвакуации для исходящего маршрута (в минутах);

n = эффективность эвакуации.

Подход предполагает, что люди следуют по направленному маршруту до места назначения, обычно на открытом воздухе или в зоне убежища. Такой направленный маршрут не учитывает возможности перехода в «неправиль-

ном» направлении (например, перемещение в кругах или блокирование дымом или огнем). По этой причине фактор эффективности обычно добавляется к моделируемому времени эвакуации.

Нет консенсуса относительно того, какое значение следует использовать для эффективности эвакуации, но 1.5 является минимальным. Для многих приложений значение эффективности 2 было бы более разумным. Например, значение 2 может быть рассмотрено для здания с двумя выходами, один из которых может быть заблокирован дымом или огнем.

4.4.2 Анализ контролируемого потока

Определим время эвакуации для нашего производственного двухэтажного склада с двумя лестницами и 50 человек на каждом этаже. Двери, ведущие в лестницы и выходящие из них, 813 миллиметра. Время предварительного движения, t_{pm} оценивается в 8 мин, а эффективность эвакуации n составляет 2.

Люди на первом этаже выходят прямо на улицу, не используя лестницу. Население оставшихся этажей с использованием двух лестниц рассчитывается как:

$1 \times 50 = 50$ человек.

Если все двери на лестничной клетке имеют одинаковую ширину, ограничение будет находиться у внешней лестничной клетки. Если ширина наружной двери меньше, чем сумма ширины внутренних дверей лестничной клетки, которые используются для эвакуации, наружная дверь все еще является ограничением.

Таблица 4.4 - Время, необходимое человеку для прохождения через внешнюю дверь лестницы

Ширина двери (мм)	Время для прохождения двери (мин)
750-800	0,03
800-850	0,025
850-900	0,02
900-950	0,015
950-1000	0,01

Учитывая, что первый человек, достигший внешней лестничной клетки, идет вниз по лестнице с пола выше, время, чтобы добраться до наружной двери, составляет около 0,5 мин.

Считается, что половина людей будет использовать одну лестницу, а половина будет использовать другую. Таким образом, население, использую-

щее лестницу, составляет P = 25. Как видно из таблицы 4.4, ß = 0,025 для наружной двери на лестничной клетке 813 миллиметра. Время прохождения населения через внешнюю лестницу оценивается как:

$$t_c = ß \times P, \tag{4.14}$$

где ß = время, когда человек проходит через внешнюю лестницу (в минутах);

P = население, эвакуирующееся по лестнице.

$t_c = 0{,}025 \times 25 = 0{,}625$ мин.

Время, необходимое человеку для прохождения через внешнюю дверь лестницы, ß, зависит от ширины двери.

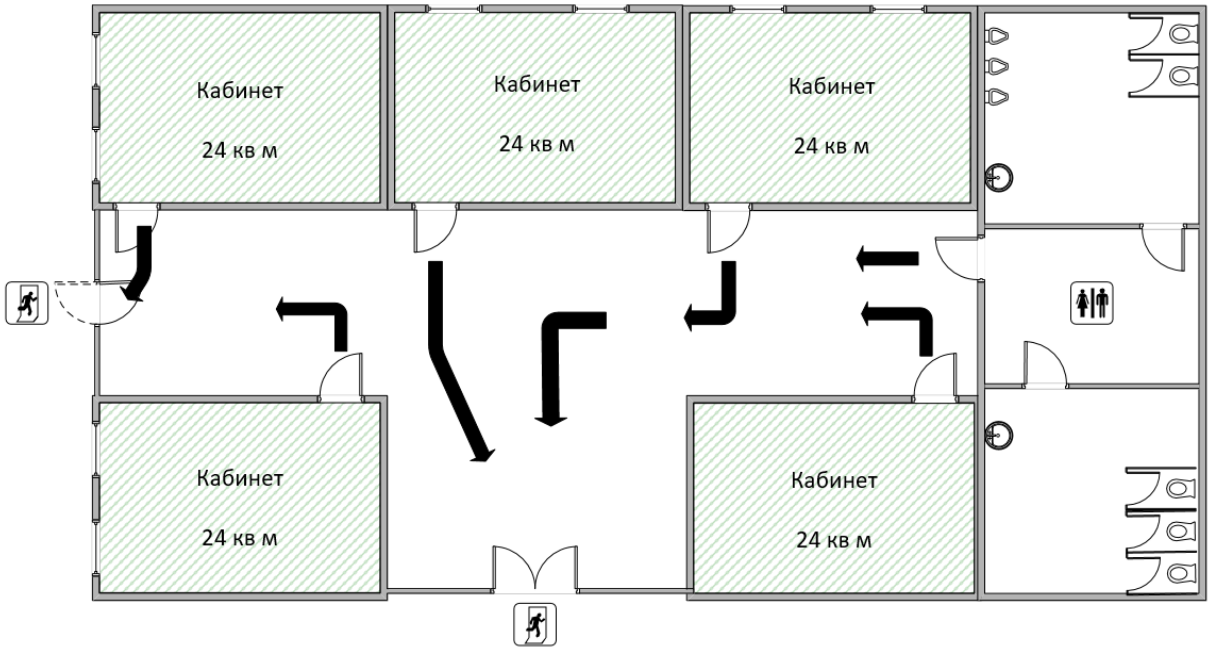


Рисунок 4.4. – План эвакуации

Подход с ограниченным потоком подходит для системы выхода, которая имеет точку, в которой формируется линия ожидающих людей. Учитывая, что ограничение является наружной дверью на лестничной клетке, и все пассажиры начинают свою эвакуацию одновременно в нулевой момент времени, время моделирования эвакуации оценивается как:

$$t_m = t_a + t_c, \tag{4.15}$$

где t_m = смоделированное время эвакуации для исходящего маршрута (в минутах);

t_a = время, когда первый человек должен прийти к внешней лестничной клетке (в минутах);

t_c = время для прохода населения через наружную лестничную клетку (в минутах).

$t_m = 0,5 + 0,625 = 1,125$ мин.

Общее время эвакуации по формуле 4.13:

$t_t = 8 + 2 \times 1,125 = 10,25$ мин.

5 Экономическое обоснование проекта

В данном дипломном проекте рассматривается разработка базы данных для производственного склада.

В современном мире, база данных востребована в любых складах для учета продуктов и товаров. Данный программный продукт, непосредственно, предоставляет данные и количество разнообразных товаров склада.

Графический интерфейс базы данных позволяет пользователю получать данные в понятном и удобном виде.

В разделе экономической части рассматривается показатели, отражающие экономические, материальные и трудовые затраты на проект.

5.1 Расчет вычисления расходов на разработку проекта

Расчет полных затрат на разработку проектного решения в виде информационных технологий $(C_{пі})$ осуществляется по формуле:

$$C_{пі} = З_{фот} + З_{сзі} + M_i + P_{сі} + П_{зі} + P_{ні}, \quad (5.1)$$

где $З_{фот}$ – общий фонд оплаты труда разработчиков, тенге;

$З_{сзі}$ – отчисления по социальному налогу, тенге;

M_i – затраты на материалы, тенге;

$P_{сі}$ – затраты на специальные программные средства, необходимые для разработки проектного решения, тенге;

$П_{зі}$ – прочие затраты, тенге;

$P_{ні}$ – накладные расходы, тенге.

Размер фонда оплаты труда разработчиков ($З_{ФОТ}$) рассчитывается по формуле:

$$З_{ФОТ} = З_{оі} + З_{ді}, \quad (5.2)$$

где $З_{оі}$ – основная заработная плата, тенге;

$З_{ді.}$ – дополнительная заработная плата, тенге.

На основе информации о разработанных программных функциях, функция каталогов определяет объем функциональности и общий объем программного обеспечения, обновленный (скорректированный) при соблюдении условий разработки программного обеспечения организации. Модифицированный объем программного обеспечения ($V_у$) рассчитывается по формуле:

$$V_у = \sum_{i=1}^{n} V_{yi}, \quad (5.3)$$

где V_{yi} − уточненный объем отдельной функции ПО (LOC).

Таблица 5.1 – Каталог функции программного обеспечения

№	Наименование (содержание) функций	Объем функций (строк исходного кода)
		С использованием среды разработки приложений
		Visual C# (Microsoft)
1	Генерация структуры базы данных	4300
2	Обработка наборов и записей базы данных	2670
3	Организация поиска и поиск в базе данных	5480
Итого		12450

Из таблицы 5.1 определяем объем ПО (строки исходного кода приложения, LOC).

Таким образом,

$V_0 = 12450$ строк кода.

Общая трудоемкость проекта рассчитывается по формуле:

$$T_o = T_н \times K_c \times K_m \times K_н , \quad (5.4)$$

где K_c – коэффициент, учитывающий сложность ПО;

K_m – поправочный коэффициент, учитывающий степень использования при разработке стандартных модулей;

$K_н$ – коэффициент, учитывающий степень новизны ПО.

Коэффициент сложности основан на данных, приведенных в таблице 5.2, а значение K_c равно 0,18, поскольку в этом документе перечислены три функции: память, работа системы, графический интерфейс.

Таблица 5.2 – Дополнительные коэффициенты сложности ПО

Характеристические данные программы	Значения K_c
две характеристики	0,12
три характеристики	0,18
свыше трех характеристик	0,26

Поправочный коэффициент, учитывающий степень использования для проектирования стандартных модулей (K_m), определяется на основе данных в таблице 5.2 и Kt = 0,7.

Таблица 5.3 – Значения поправочного коэффициента, учитывающего использование стандартных модулей типовых программ и ПО ($К_т$)

Степень охвата реализуемых функций разрабатываемого ПО стандартными модулями, типовыми программами и ПО	Значение
От 40 % до 60	0,7

Поправочный коэффициент, определяющий новизну этого проекта, рассчитывается с использованием данных в таблице 5.4 и $К_Н = 0,9$.

Таблица 5.4 – Поправочные коэффициенты (Кн)

Категория новизны	Степень новизны	Использование		Значение $К_н$
		На основе нового типа ПО	В среде новой ОС	
А	Принципиально новые приложения, которые не имеют аналогов	+	+	1,75
		-	+	1,6
		+	-	1,2
		-	-	1,0
Б	Приложения, которые являются развитием определенного параметрического ряда ПО	+	+	1,0
		-	-	0,9
		+	-	0,8
В	ПО, являющиеся развитием определенного параметрического ряда ПО, разработанных для ранее освоенных типов конфигурации ПК и ОС	-	-	0,7

Наилучшие временные стандарты для разработки этого дипломного проекта, в зависимости от команды сложности и указанного набора программного обеспечения, являются основой для поиска нормативной сложности.

Основываясь на объеме программного интерфейса и категории сложности, который определяет новый объект и сложность этого проекта и на уровне использования его стандартных модулей разработки, определяется нормативный объем создания $Т_п$ (таблица 5.5).

Таблица 5.5 – Классификация типов программного обеспечения

Категория сложности	Характеристика ПО
II	Обеспечение настройки ПО на изменения структур входных и выходных данных.

Таблица 5.6 - Укрупненные нормы времени на разработку ПО ($Т_н$) в зависимости от уточненного объема ПО ($V_у$) и группы сложности ПО (чел./дн.)

Объем ПО (строки исходного кода, LOC)	Категории сложности ПО	
	I	II
11000	349	291
12000	374	312
13000	399	333
14000	427	356

Для II категории сложности приложения:

$Т_н = 312$.

Общий трудовой объем осуществляется по формуле (5.4):

$Т_о = 312 \cdot 0,18 \cdot 0,7 \cdot 0,9 = 35,38$ (чел./час.).

Планируемое количество сотрудников $С_{hr}$ и запланированные периоды времени, необходимые для разработки этого общего проекта, определяются на основе работы. И следующие задачи могут быть выполнены:

– расчет численности работников на указанную дату реализации проекта;

– расчет времени реализации проекта для определенного количества сотрудников.

Численность работников проекта $Ч_р$ определяется по формуле:

$$Ч = Т_о / (Т_р \cdot Ф_{эф}), \quad (5.5)$$

где $Ф_{эф}$ – эффективный фонд времени работы одного работника в течение года (дн.);

$Т_о$ – общая трудоемкость реализации программного интерфейса (чел./дн.);

$Т_р$ – срок создания проекта (лет).

Срок создания проекта ($Т_р$) вычисляется по формуле:

$$Т_р = Т_о / (Ч_р \cdot Ф_{эф}), \quad (5.6)$$

где $Ч_р$ – плановое число работников.

Эффективный фонд времени работы одного работника ($Ф_{эф}$) вычисляется по формуле:

$$Ф_{эф} = Д_г – Д_в - Д_о – Д_п, \quad (5.7)$$

где $Д_г$ – общее число дней работника в году;
Дп – общее число праздничных дней работника в году;
Дв – общее число выходных дней работника в году;
До – общее число дней отпуска работника.

Так как, в соответствии с производственным календарем на 2018 год:

$Д_о = 24$, $Д_п = 15$, $Д_г = 365$, $Д_в = 101$.

Эффективный фонд времени одного работника составит:

$Ф_{эв} = 365 - 101 - 24 - 15 = 225$ дней.

Плановое общее число работников $Ч_р = 1$, следовательно, по формуле (5.6)

$Т_р = 35{,}38 / (1 \times 225) = 0{,}157$ лет = 57 дней.

Тогда плановое количество работников

Ч = 35,38 / (0,157 * 225) = 1 чел.

Основной заработок на конкретную программу осуществляется по формуле:

$$З_{оi} = \sum_{i=1}^{n} Т_{чi} \times Т_ч \times Ф_п \times К, \quad (5.8)$$

где n − количество исполнителей, занятых разработкой конкретного ПО;

$Т_{чi}$ − часовая тарифная ставка i-го исполнителя (тыс.тенге);
$Ф_п$ − плановый фонд рабочего времени i-го исполнителя (дней);

$T_ч$ − количество часов работы в день (час);
K − коэффициент премирования.

Поскольку создание этого проекта займет около 57 дней.

План персонала группы сотрудников, участвующих в проекте, и определяющих подготовку, квалификацию, специализацию и должность, основывается на данных подтверждения.

В зависимости от специфики и сложности выполняемых функций команда сотрудников специализированных команд, участвующих в разработке программного обеспечения, с определением обучения, специализации, квалификации и должности (таблица 5.7).

Таблица 5.7 - Сведения по работникам, участвующих в данном проекте

Разработчик	Количество, чел	Заработок в месяц, тенге
Специалист по БД	1	250000
Итого	1	250000

Почасовая ставка рассчитывается путем деления месячной зарплаты на основе 40-часового рабочего дня еженедельника и общего времени ($Ф_р$):

$$Т_ч = \frac{Т_м}{Ф_р}, \quad (5.9)$$

где $Т_м$ − месячная тарифная ставка (тенге).
$Т_ч$ − часовая тарифная ставка (тенге);

Формула вычисления общего фонда времени

$$Ф_р = Т_ч \cdot Ф_п \quad (5.10)$$

Таким образом, получается

$Ф_р = 1488 \times 57 \times 8 = 678528$ тенге.

Тарифная ставка специалиста по БД проекта:

$Т_ч = 250000 / 168 = 1488$ тенге/час.

Основная заработная плата специалиста на конкретное ПО составит:

$З_{оі} = 1488 \times 1{,}38 \times 8 \times 57 = 936369$ тенге.

Итоги расчета основного заработка записаны в таблицу 5.8.

Таблица 5.8 – Итоги расчета расхода основной заработной платы

Наименование содержания работ	Разработчик	Трудовой объем, час	Заработная плата за час работы, т/час	Сумма заработной платы, тенге
Тех. задание	Специалист по БД	40	1488	59520
Моделирование	Специалист по БД	40	1488	59520
Программирование	Специалист по БД	240	1488	357120
Тестирование	Специалист по БД	80	1488	119040
Внедрение	Специалист по БД	56	1488	83328
Итого		456	1488	678528

Дополнительная заработная плата составляет 10% от основной заработной платы и вычисляется по формуле:

$$З_{дi} = З_{оi} \cdot Н_д / 100, \quad (5.11)$$

где $Н_д$ – коэффициент дополнительного заработка работника.

$З_{дi} = 936369 \cdot 10 / 100 = 93637$ тенге.

Общий фонд оплаты труда работника равен

$З_{фот} = 936369 + 93637 = 1\ 030\ 006$ тенге.

По ст. 358 п. 1 НК РК социальный налог ($З_{сзi}$) составляет 11 % от дохода работника, и вычисляется по формуле:

$$З_{сзi} = (З_{фот} – ПО) \cdot 11\%, \quad (5.12)$$

где ПО – пенсионные отчисления, составляющий 10% от ФОТ.

$$ПО = З_{фот} \cdot 10\ \% \quad (5.13)$$

Вычисление пенсионного отчисления:

$ПО = 1\ 030\ 006 \cdot 0{,}1 = 103000$ тенге,

$З_{сзі}$ = (1 030 006 – 103000) · 0,11 = 101 970,7 тенге.

Расходы на материалы находятся по формуле:

$$М_і = (З_{осн} \cdot Н_{мз}) / 100\%, \quad (5.14)$$

где $Н_{мз}$ − норма расхода материалов от основного заработка (3–5%).

$М_і$ = 936369 · 0,04 = 37 454,8 тенге.

В стоимость «Специального оборудования и программного обеспечения» входит стоимость средств. Приобретение дополнительного программного обеспечения и ПО для разработки специального программного интерфейса, включая проектирование, производство, устранение неполадок, установку и эксплуатацию вычисляется по формуле:

$$Р_{сі} = \sum_{i=1}^{n} Ц_{ci} \quad (5.15)$$

Таблица 5.9 – Перечень оборудования, необходимого для разработки

Наименование	Характеристика	Количество единиц	Цена, (тенге)	Общая сумма, (тенге)
Ноутбук	Asus Vivobook	1	190000	190000
Итого				190000

Таблица 5.10 – Программное обеспечение, используемое в работе

Программное обеспечение	Стоимость, тенге
Oracle PL/SQL	бесплатно
XAMPP ControlPanel	бесплатно
ERwin Data Modeler r7	бесплатно
Итого	0

Расходы по прочим затратам составляют:

$Р_{сі}$ = 190000 тенге.

«Другие издержки» для конкретного приложения включают, но не ограничиваются, затраты на приобретение и подготовку специальной научно-технической информации и литературы. Они основаны на стандарте для организации в процентах от основной заработной платы:

$$П_{зі} = З_{оі} \cdot Н_{пз} / 100, \quad (5.16)$$

где $Н_{пз}$ − норматив прочих расходов в общем по организации в (20%).

Таким образом:

$П_{зі}$ = 936369 · 0,2 = 187 273,8 тенге.

Затраты части «Накладные расходы» $Р_{нi}$ вычисляется по нормативу $Н_{рн}$.

В процентном отношении к основной заработной плате работника. Норматив устанавливается в целом по организации

$$Р_{нi} = З_{оi} \cdot Н_{рн}/100\%, \quad (5.17)$$

где $Р_{нi}$ – накладные расходы на определенное ПО (тыс.тенге);
$Н_{рн}$ – норматив накладных расходов в целом по организации 70%.

$Р_{нi}$ = 936369 · 0,7 = 655 458,3 тенге.

Полные затраты на создание базы данных, формула (5.1):

$С_{пi}$ = 1 030 006 + 101 970,7 + 37 454,8 + 190000 + 187 273,8 + 655 458,3 = 2 202 163,6 тенге.

Результаты расчета расходов на создание БД и их структура представлены в таблице 5.11 и на рисунке 5.1.

Таблица 5.11 – Итоги расчета расходов на разработку базы данных

Затраты на разработку	Условное обозначение	Значение, тенге	В процентах от общей суммы
Фонд оплаты труда	Зфот	1 030 006	46,77
Социальный налог	Зсзі	101 970,7	4,63
Материалы	$М_i$	37 454,8	1,70
Прочие затраты	$П_{зі}$	187 273,8	8,50
Специальные оборудования и ПО	$Р_{сі}$	190000	8,63
Накладные расходы	Рні	655 458,3	29,76
Итого		2 202 163,6	100

Рисунок 5.1. – Структура расходов на создание базы данных с графическим интерфейсом

5.2 Расчет цены программного продукта

Расчет цены программного продукта, который был реализован одной организаций по заказу другой организацией, а также не предназначенный для тиража вычисляется по формуле:

$$Ц_{пп} = З_{рпр} + П_{п} + НДС, \quad (5.18)$$

где $Ц_{пп}$ – цена программного продукта, тенге;

$П_{п}$ – планируемая прибыль, тенге;

$З_{рпр}$ – затраты на реализацию проекта, в данном случае программного интерфейса, тенге;

НДС – налог на добавленную стоимость, тенге.

Планируемая прибыль составляет 20% от себестоимости разработки.

$П_{п} = 2\ 202\ 163{,}6 \times 0{,}2 = 440\ 432{,}7$ тенге.

НДС, начисленный на программу, осуществляется следующим образом:

$$НДС = (З_{рпр} + П_{п}) \cdot k_{НДС}, \quad (5.19)$$

где $k_{НДС}$ – ставка налога на добавленную стоимость.

Подставив данные в формулу 5.19, получаем

НДС = (2 202 163,6 + 440 432,7) · 0,12 = 317 111,6 тенге.

Подставив данные в формулу 5.18, получаем

$Ц_{пп}$ = 2 202 163,6 + 440 432,7 + 317 111,6 = 2 959 707,8 тенге.

5.3 Вывод по технико–экономической части

Разработка базы данных для производственного склада является дорогостоящим проектом, требующих существенных затрат, а также обязательного использования компьютерной техники и программного оснащения. В экономической части дипломного проекта был проведен расчет издержек на создание базы данных производственного склада и расчет стоимости программного продукта. Стоимость разработки базы данных производственного склада с графическим интерфейсом в программе Visual Studio 2015 составляет 2 202 163,6 тенге.

Заключение

В этом дипломном проекте проделана работа над проектированием и разработкой базы данных и графического интерфейса пользователя для производственного склада. Была проделана работа по разработке структуры БД, а также реализовано подключение графического интерфейса к самой базе данных. Различные программные средства реализованы для разработки базы данных и ее интерфейса, они являются наиболее удобными и актуальными, это Oracle Database Server и объектно-ориентированный язык программирования на C #.

Программный интерфейс сконструирован для использования на коммерческой основе. На сегодняшний день множество складов стремятся получить прибыль, предоставляя свои услуги. В экономической части было показано, что данный проект является актуальным, потому что он позволяет увеличить доход предприятия. Цена реализации базы данных производственного склада составит 2 959 707,8 тенге. Создание данного программного продукта является экономически выгодным и эффективным.

В разделе безопасности жизнедеятельности произведен анализ условий труда, расчет систем пожаротушения, кондиционирования и эвакуационных путей.

В результате написания программы, однопользовательское приложение имеет удобный, интуитивно понятный интерфейс, сохраненную систему хранения данных и возможность расширения базы данных.

Список литературы

1 Банник М. MySQL 2011, Справочник БД. – 1-е изд. – СПб: Москва, 2012. – 417 с.

2 Сноу Г., Стивен П. Структура баз данных: разработка и управление. – 4-е изд. – Спб.: Санкт-Петербург, 2005. – 927 с.

3 Сойер Дж., Гендэрлоф А. Язык программирования C# .NET. Руководство по применению. – КОРОНА принт, 2007. – 911 с.

4 Ваденова С. М. C++ в среде разработки Visual Studio NET. Руководство для программиста.– М.: КУДИС-ОБРЗ, 2004. – 358с.

5 Мюллер К. Освоение UML - диаграмм. [Текст]: М.: Издательский дом «Вильнюс», 2008. – 428 с.

6 UML - диаграммы. Лекции. [Электронный ресурс] – Режим доступа: http://www.intuit.ru.-28.09.2011.

7 Г.М. Гукаян. Экономика в производственных компаниях: Справочник. – М.: ИНФА-МСК, 2009. – 480 с.

8 Экономика предприятия: руководство / А.Е. Петров. – М.: ИНФА-МСК, 2012. – 366 с.

9 Д.М. Писаренкоо. Экономическое обоснование: учебное пособие. – М.: ИЦ- РИОР, 2011. – 157 с.

10 А.И. Рой. Экономика: методические указания. – М.: КРс, 2013. – 401 с.

11 Методические указания к выполнению экономической части дипломных работ для студентов специальности 5В070400– Вычислительная техника и программное обеспечение / Еркешева З.Д., Г.Ш.Боканова. – Алматы: АУЭС, 2013. – 40с.

12 Чернов С.В., Ильницкий А.В., Козьянов А.Ф. БЖД. Учебник для ВУЗов. – М.: Новый раздел, 2007. – 513 с.

13 СНиП РК 4.02–42–2006 Отопление, вентиляция и кондиционирование. – Астана: Издательство стандартов, 2007.

14 СНиП РК 2.02–15–2003 Пожарная автоматика зданий и сооружений. – Астана: Издательство стандартов, 2004.

15 Шилдт Герберт - C# 4.0 полное руководство – 2011.

16 Oracle Database 10g. Руководства баз данных для предприятия, Алипади Грэм Р, 2009.

17 https://docs.oracle.com/cloud/latest/db112/CNCPT/tablecls.htm#CNCPT113

18 https://docs.oracle.com/cloud/latest/db112/CNCPT/intro.htm#CNCPT939

Приложение А. Листинг программы

(обязательно)

```
CREATE  TABLE  sklad (
            sklad_id              number(10)   NOT NULL PRIMARY KEY,
   sklad_name           varchar2(30)  NOT NULL,
            sklad_address          varchar2(70)  NOT NULL,
            telephone             numeric(10),
            sklad_city            varchar2(20));
CREATE  TABLE producer (
            prod_name                 varchar2(15) PRIMARY KEY,
            cust_country              varchar2(20));
CREATE  TABLE customer (
            cust_name                 varchar2(15) PRIMARY KEY,
            cust_add            varchar2(25),
            cust_city                 varchar2(20),
            cust_tel                  numeric(10));
CREATE  TABLE product (
   prod_id              number(10)  NOT NULL PRIMARY KEY,
            prod_data                   date,
   prod_name                varchar2(15) REFERENCES producer(prod_name),
   cust_name                varchar2(15) REFERENCES customer(cust_name),
   sklad_id             number(10) REFERENCES sklad(sklad_id));
CREATE  TABLE  appliances (
   app_name                varchar2 (20) NOT NULL PRIMARY KEY,
   prod_id                 number(10)  NOT NULL REFERENCES product
(prod_id ),
   app_model               varchar2 (20),
            app_data                    date,
            app_weight                  number(5),
            app_sent                    date,
            app_number                  number(10)) ;
CREATE  TABLE  smartphone (
   smart_name           varchar2 (20),
   smart_id             number(10)  NOT NULL PRIMARY KEY,
            prod_id                 number(10)  NOT NULL REFERENCES product
(prod_id ),
   smart_model          varchar2 (20),
            smart_data                  date,
            smart_weight            number(5),
            smart_sent                  date,
            smart_number            number(10)) ;
```

```
CREATE  TABLE  laptop (
   laptop_name          varchar2 (20),
   laptop_id            number(10)  NOT NULL PRIMARY KEY,
            prod_id                 number(10)  NOT NULL REFERENCES product
(prod_id ),
   laptop_model         varchar2 (20),
            laptop_data                 date,
            laptop_weight           number(5),
            laptop_sent                 date,
            laptop_number           number(10)) ;
CREATE  TABLE personal (
            sklad_id                number(10) REFERENCES sklad(sklad_id),
            pers_id                     number(10)  NOT NULL,
   pers_fio             varchar2 (20)  NOT NULL,
            pers_addrs                  varchar2 (70),
            pers_tel              numeric(10),
            pers_exp                    number(10),
            pers_data                   date,
   CONSTRAINT PK_sklad PRIMARY KEY (pers_id ));
CREATE  TABLE position (
            pers_id                     number(10)  NOT NULL,
            post_dolj                   varchar2 (20),
            post_zarp                   number(20),
            post_data                   date,
   CONSTRAINT FK_position_personal FOREIGN KEY (pers_id ) REFER-
ENCES personal (pers_id ),
            CONSTRAINT  PK_position PRIMARY KEY ( pers_id ));
CREATE  TABLE transport (
            sklad_id                number(10) REFERENCES sklad(sklad_id),
            tran_id                     number(10)  NOT NULL PRIMARY KEY,
   tran_name            varchar2 (20)  NOT NULL,
            tran_model                  varchar2 (20),
            tran_number                 number(15),
            tran_data                   date);
   commit;
INSERT INTO  sklad(sklad_id,sklad_name,sklad_address,telephone,sklad_city)
VALUES(1, 'Electronics Almaty','Berdigulova 130',7473736052,'Almaty');
INSERT INTO  sklad(sklad_id,sklad_name,sklad_address,telephone,sklad_city)
VALUES(2, 'Electronics Taldykorgan','Masanchi 72',7473736053,'Taldykorgan');
INSERT INTO  sklad(sklad_id,sklad_name,sklad_address,telephone,sklad_city)
VALUES(3, 'Electronics Astana','Amangeldi 6',7473736054,'Astana');
INSERT INTO  sklad(sklad_id,sklad_name,sklad_address,telephone,sklad_city)
VALUES(4, 'Electronics Pavlodar','Baytursinova 127',7473736055,'Pavlodar');
```

```
INSERT INTO  sklad(sklad_id,sklad_name,sklad_address,telephone,sklad_city)
VALUES(5, 'Electronics Aktau','Gvardeyskaya 77',7473736051,'Aktau');
commit;
INSERT INTO  producer(prod_name,cust_country)
VALUES('Samsung','Korea');
INSERT INTO  producer(prod_name,cust_country)
VALUES('LG','Japan');
INSERT INTO  producer(prod_name,cust_country)
VALUES('Huawei','China');
INSERT INTO  producer(prod_name,cust_country)
VALUES('Asus','USA');
INSERT INTO  producer(prod_name,cust_country)
VALUES('Fujitsu','Germany');
commit;
INSERT INTO  customer(cust_name,cust_add,cust_city,cust_tel)
VALUES('Technodom','Aynabulak 3','Almaty',7082202220);
INSERT INTO  customer(cust_name,cust_add,cust_city,cust_tel)
VALUES('Sulpak','Gogolya 17','Kizilorda',7083303330);
INSERT INTO  customer(cust_name,cust_add,cust_city,cust_tel)
VALUES('Mechta','Abaya 120','Semey',7084404440);
INSERT INTO  customer(cust_name,cust_add,cust_city,cust_tel)
VALUES('Electronika','Derjinskaya 201','Novgorod',7085505550);
INSERT INTO  customer(cust_name,cust_add,cust_city,cust_tel)
VALUES('Eldarado','Shumnaya 321','Kiev',7086606660);
commit;
INSERT INTO  product(prod_id,prod_data,prod_name,cust_name,sklad_id)
VALUES(101,'22.08.2018','Samsung','Technodom',1);
INSERT INTO  product(prod_id,prod_data,prod_name,cust_name,sklad_id)
VALUES(102,'22.03.2018','LG','Sulpak',2);
INSERT INTO  product(prod_id,prod_data,prod_name,cust_name,sklad_id)
VALUES(103,'30.05.2018','Huawei','Mechta',3);
INSERT INTO  product(prod_id,prod_data,prod_name,cust_name,sklad_id)
VALUES(104,'15.09.2018','Asus','Electronika',4);
INSERT INTO  product(prod_id,prod_data,prod_name,cust_name,sklad_id)
VALUES(105,'10.04.2018','Fujitsu','Eldarado',5);
commit;
INSERT INTO  applianc-
es(app_name,prod_id,app_model,app_data,app_weight,app_sent,app_number)
VALUES('TV',101,'Ultra HD','12.03.2015',12,'25.05.2019',40);
INSERT INTO  applianc-
es(app_name,prod_id,app_model,app_data,app_weight,app_sent,app_number)
VALUES('Holodilnik',101,'SuperCold','01.07.2014',50,'05.05.2019',20);
```

```
INSERT INTO  appliances(app_name,prod_id,app_model,app_data,app_weight,app_sent,app_number)
VALUES('Stirlnaya mashina',102,'BubbleWash','21.10.2016',35,'01.11.2018',25);
INSERT INTO  appliances(app_name,prod_id,app_model,app_data,app_weight,app_sent,app_number)
VALUES('Plita',102,'AH531Wa','24.04.2017',50,'31.03.2019',30);
INSERT INTO  appliances(app_name,prod_id,app_model,app_data,app_weight,app_sent,app_number)
VALUES('Mikrovolnovka',101,'MicroVivo','17.06.2017',15,'18.01.2019',40);
commit;
INSERT INTO smartphone(smart_name,smart_id,prod_id,smart_model,smart_data,smart_weight,smart_sent,smart_number)
VALUES('Galaxy',3736050,101,'S9','19.09.2017',1,'18.01.2019',40);
INSERT INTO smartphone(smart_name,smart_id,prod_id,smart_model,smart_data,smart_weight,smart_sent,smart_number)
VALUES('K10 LTE',3736051,102,'Black blue','24.10.2017',1,'20.02.2019',30);
INSERT INTO smartphone(smart_name,smart_id,prod_id,smart_model,smart_data,smart_weight,smart_sent,smart_number)
VALUES('X Power',3736052,102,'DS LTE','19.12.2017',1,'10.01.2019',45);
INSERT INTO smartphone(smart_name,smart_id,prod_id,smart_model,smart_data,smart_weight,smart_sent,smart_number)
VALUES('Mate 10',3736053,103,'Pro 128GB','12.12.2017',1,'18.02.2019',35);
INSERT INTO smartphone(smart_name,smart_id,prod_id,smart_model,smart_data,smart_weight,smart_sent,smart_number)
VALUES('Zenfone',3736054,104,'RedEdition','30.11.2017',1,'01.02.2019',20);
commit;
INSERT INTO  laptop(laptop_name,laptop_id,prod_id,laptop_model,laptop_data,laptop_weight,laptop_sent,laptop_number)
VALUES('Vivobook',4837090,104,'SuperBattery','27.11.2017',8,'01.02.2019',20);
INSERT INTO  laptop(laptop_name,laptop_id,prod_id,laptop_model,laptop_data,laptop_weight,laptop_sent,laptop_number)
VALUES('Lifebook',4837091,105,'AH531','19.12.2017',6,'01.02.2019',17);
INSERT INTO  laptop(laptop_name,laptop_id,prod_id,laptop_model,laptop_data,laptop_weight,laptop_sent,laptop_number)
VALUES('Prestigio',4837092,102,'MultiPad','29.11.2017',7,'02.02.2019',15);
```

```
INSERT INTO  lap-
top(laptop_name,laptop_id,prod_id,laptop_model,laptop_data,laptop_weight,laptop
_sent,laptop_number)
VALUES('Switch',4837093,105,'Aspire One','27.12.2017',5,'17.01.2019',20);
commit;
INSERT INTO  person-
al(sklad_id,pers_id,pers_fio,pers_addrs,pers_tel,pers_exp,pers_data)
VALUES(1,1001,'Hashimjan D.A.','Zarya-Vostoka',7028423692,4,'24.07.1994');
INSERT INTO  person-
al(sklad_id,pers_id,pers_fio,pers_addrs,pers_tel,pers_exp,pers_data)
VALUES(1,1002,'Ilbesinov N.O.','Atakent',7009643623,5,'15.11.1993');
INSERT INTO  person-
al(sklad_id,pers_id,pers_fio,pers_addrs,pers_tel,pers_exp,pers_data)
VALUES(1,1003,'Toimov A.J.','Almalinka',7077001776,4,'11.07.1996');
INSERT INTO  person-
al(sklad_id,pers_id,pers_fio,pers_addrs,pers_tel,pers_exp,pers_data)
VALUES(1,1004,'Smakotin A.P.','Almaty 1',7077402614,3,'04.03.1997');
INSERT INTO  person-
al(sklad_id,pers_id,pers_fio,pers_addrs,pers_tel,pers_exp,pers_data)
VALUES(1,1005,'Kim A.A.','Rabochiy',7018087900,2,'14.10.1996');
commit;
INSERT INTO  position(pers_id,post_dolj,post_zarp,post_data)
VALUES(1001,'Director',500000,'24.07.2014');
INSERT INTO  position(pers_id,post_dolj,post_zarp,post_data)
VALUES(1002,'IT',200000,'03.10.2015');
INSERT INTO  position(pers_id,post_dolj,post_zarp,post_data)
VALUES(1003,'Buhgalter',250000,'15.05.2013');
INSERT INTO  position(pers_id,post_dolj,post_zarp,post_data)
VALUES(1004,'IT',200000,'30.07.2014');
INSERT INTO  position(pers_id,post_dolj,post_zarp,post_data)
VALUES(1005,'Voditel',120000,'19.09.2015');
commit;
INSERT INTO
transport(sklad_id,tran_id,tran_name,tran_model,tran_number,tran_data)
VALUES(1,3333,'Mersedes','G250',570570,'01.02.2017');
INSERT INTO
transport(sklad_id,tran_id,tran_name,tran_model,tran_number,tran_data)
VALUES(1,3331,'Gazel','G23TR',570571,'21.07.2015');
INSERT INTO
transport(sklad_id,tran_id,tran_name,tran_model,tran_number,tran_data)
VALUES(1,3332,'Kamaz','K789ZE',570572,'18.10.2014');
commit;
```

Приложение Б. Листинг программы

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.OleDb;

namespace WindowsFormsApplication3
{
   public partial class Form1 : Form
   {
      Button b1, b2;
      TextBox t1, t2;
      Label l1, l2;
      public Form1()
      {
         Text = "Авторизация"; Width = 600; Height = 300; this.BackgroundImage = new Bit-
map(@"C:/oracle/sklad.jpg");

         b1 = new Button(); b1.Text = "Вход"; b1.Location = new Point(230, 155); b1.Width = 140; b1.Height = 30;
Controls.Add(b1); b1.Click += b1_click;
         b2 = new Button(); b2.Text = "Выход"; b2.Location = new Point(420, 220); b2.Width = 140; b2.Height = 30;
Controls.Add(b2); b2.Click += b2_click;

         t1 = new TextBox(); t1.Location = new Point(230, 115); t1.Width = 140; t1.Height = 30; Controls.Add(t1);
         t2 = new TextBox(); t2.Location = new Point(230, 85); t2.Width = 140; t2.Height = 30; Controls.Add(t2);

         l2 = new Label(); l2.Text = "Login"; l2.Location = new Point(130, 85); l2.Width = 60; l2.Height = 20; Con-
trols.Add(l2);
         l1 = new Label(); l1.Text = "Password"; l1.Location = new Point(130, 115); l1.Width = 80; l1.Height = 20;
Controls.Add(l1);
      }

      private void b2_click(object sender, EventArgs e)
      {
         Application.Exit();
      }

      private void b1_click(object sender, EventArgs e)
      {
         if (t1.Text == "admin" & t2.Text == "admin")

         {
            MessageBox.Show("Вход выполнен");
            this.Hide(); new Form2(this).Show(); Visible = false;
         }

         else MessageBox.Show("Введены неправильные данные");

      }

      class Form2 : Form
```

```
{
        Form1 pf;
        Button btn1, btn2, btn3, btn4, btn5, btn6, btn7, btn8;
        public Form2(Form1 f)
        {
           pf = f; Text = "Главная форма"; Width = 380; Height = 300;

           btn1 = new Button(); btn1.Text = "Склады"; btn1.Location = new Point(70, 40); btn1.Width = 80;
btn1.Height = 30; Controls.Add(btn1); btn1.Click += btn1_click;

           btn2 = new Button(); btn2.Text = "Заказчики"; btn2.Location = new Point(70, 90);
           btn2.Width = 80; btn2.Height = 30; Controls.Add(btn2); btn2.Click += btn2_click;

           btn3 = new Button(); btn3.Text = "Бытовая техника"; btn3.Location = new Point(40, 140); btn3.Width =
140;
           btn3.Height = 30; Controls.Add(btn3); btn3.Click += btn3_click;

           btn4 = new Button(); btn4.Text = "Смартфоны"; btn4.Location = new Point(70, 190); btn4.Width = 80;
           btn4.Height = 30; Controls.Add(btn4); btn4.Click += btn4_click;

           btn5 = new Button(); btn5.Text = "Ноутбуки"; btn5.Location = new Point(200, 40); btn5.Width = 80;
           btn5.Height = 30; Controls.Add(btn5); btn5.Click += btn5_click;

           btn6 = new Button(); btn6.Text = "Сотрудники"; btn6.Location = new Point(200, 90); btn6.Width = 80;
           btn6.Height = 30; Controls.Add(btn6); btn6.Click += btn6_click;

           btn7 = new Button(); btn7.Text = "Транспорт"; btn7.Location = new Point(200, 140); btn7.Width = 80;
           btn7.Height = 30; Controls.Add(btn7); btn7.Click += btn7_click;

           btn8 = new Button(); btn8.Text = "Выход"; btn8.Location = new Point(240, 220); btn8.Width = 100;
           btn8.Height = 30; Controls.Add(btn8); btn8.Click += btn8_click;
        }

        private void btn8_click(object sender, EventArgs e)
        {
           Application.Exit();
        }

        private void btn7_click(object sender, EventArgs e)
        {
           this.Hide(); new Form9(this).Show(); Visible = false;
        }

        private void btn6_click(object sender, EventArgs e)
        {
           this.Hide(); new Form8(this).Show(); Visible = false;
        }

        private void btn5_click(object sender, EventArgs e)
        {
           this.Hide(); new Form7(this).Show(); Visible = false;
        }

        private void btn4_click(object sender, EventArgs e)
        {
           this.Hide(); new Form6(this).Show(); Visible = false;
        }

        private void btn3_click(object sender, EventArgs e)
        {
```

```
this.Hide(); new Form5(this).Show(); Visible = false;
        }

        private void btn2_click(object sender, EventArgs e)
        {
            this.Hide(); new Form4(this).Show(); Visible = true;
        }

        private void btn1_click(object sender, EventArgs e)
        {
            this.Hide(); new Form3(this).Show(); Visible = true;
        }
    }
    class Form3 : Form
    {
        Form2 pf;
        Button bt1, bt2, bt3, bt4, bt5;
        TextBox tb1;
        Label l1;
        DataGridView dgv;
        public Form3(Form2 f)
        {
            pf = f; Text = "Склады"; Width = 700; Height = 350;


            dgv = new DataGridView(); dgv.Location = new Point(150, 0); dgv.Width = 550; dgv.Height = 250; Con-
trols.Add(dgv);

            bt1 = new Button(); bt1.Text = "Просмотреть таблицу"; bt1.Location = new Point(0, 0); bt1.Width = 140;
            bt1.Height = 30; Controls.Add(bt1); bt1.Click += bt1_click;

            bt2 = new Button(); bt2.Text = "Добавить склад"; bt2.Location = new Point(0, 50); bt2.Width = 140;
bt2.Height = 30;
            Controls.Add(bt2); bt2.Click += bt2_click;

            l1 = new Label(); l1.Text = "Введите ID"; l1.Location = new Point(0, 90); l1.Width = 80; l1.Height = 20;
Controls.Add(l1);
            tb1 = new TextBox(); tb1.Location = new Point(0, 110); tb1.Width = 80; tb1.Height = 30; Con-
trols.Add(tb1);
            bt3 = new Button(); bt3.Text = "Удалить по ID"; bt3.Location = new Point(0, 140); bt3.Width = 140;
bt3.Height = 30; Controls.Add(bt3); bt3.Click += bt3_click;
            bt4 = new Button(); bt4.Text = "Поиск по ID"; bt4.Location = new Point(0, 180); bt4.Width = 140;
bt4.Height = 30; Controls.Add(bt4); bt4.Click += bt4_click;
            bt5 = new Button(); bt5.Text = "Закрыть"; bt5.Location = new Point(550, 255); bt5.Width = 80; bt5.Height
= 30; Controls.Add(bt5); bt5.Click += bt5_click;
        }

        private void bt5_click(object sender, EventArgs e)
        {
            Close();
        }

        private void bt4_click(object sender, EventArgs e)
        {
            OleDbConnection con = new OleDbConnection("Provider=MSDAORA;Data Source=orcl;Persist Security
Info=True;Password=whouse;User ID=whouse");
            con.Open();

            OleDbDataAdapter oda = new OleDbDataAdapter("select * from sklad where sklad_id =" + tb1.Text, con);
            DataTable dt = new DataTable();
```

```
            dgv.DataSource = dt;
            oda.Fill(dt);

            con.Close();
        }

        private void bt3_click(object sender, EventArgs e)
        {
            OleDbConnection con = new OleDbConnection("Provider=MSDAORA;Data Source=orcl;Persist Security
Info=True;Password=whouse;User ID=whouse");
            con.Open();
            OleDbCommand cmd = new OleDbCommand("delete from sklad where sklad_id =" + tb1.Text, con);
            cmd.ExecuteNonQuery();
            con.Close();
            bt3_click(null, null);
            {
                MessageBox.Show("Запись удалена!", "Удаление прошло успешно");
            }
            throw new NotImplementedException();
        }

        private void bt2_click(object sender, EventArgs e)
        {
            this.Hide(); new Form10(this).Show(); Visible = false;
        }
        private void bt1_click(object sender, EventArgs e)
        {
            OleDbConnection con = new OleDbConnection("Provider=MSDAORA;Data Source=orcl;Persist Security
Info=True;Password=whouse;User ID=whouse");
            con.Open();

            OleDbDataAdapter oda = new OleDbDataAdapter("select * from sklad", con);
            DataTable dt = new DataTable();

            dgv.DataSource = dt;
            oda.Fill(dt);

            con.Close();
        }
    }
    class Form10 : Form
    {
        Form3 pf;
        Button b;
        Label l1, l2, l3, l4, l5;
        TextBox t1, t2, t3, t4, t5;
        OleDbCommand cmd;
        OleDbConnection con;
        public Form10(Form3 f)
        {
            pf = f; Text = "Форма для добавления"; Width = 220; Height = 380;
            con = new OleDbConnection("Provider=MSDAORA;Data Source=orcl;Persist Security In-
fo=True;Password=whouse;User ID=whouse");

            b = new Button(); b.Text = "Добавить"; b.Location = new Point(40, 290); b.Width = 80; b.Height = 30;
Controls.Add(b);
            b.Click += b_click;

            t1 = new TextBox(); t1.Location = new Point(40, 40); t1.Width = 140; t1.Height = 30; Controls.Add(t1);
            t2 = new TextBox(); t2.Location = new Point(40, 90); t2.Width = 140; t2.Height = 30; Controls.Add(t2);
```

```
t3 = new TextBox(); t3.Location = new Point(40, 140); t3.Width = 140; t3.Height = 30; Controls.Add(t3);
            t4 = new TextBox(); t4.Location = new Point(40, 190); t4.Width = 140; t4.Height = 30; Controls.Add(t4);
            t5 = new TextBox(); t5.Location = new Point(40, 240); t5.Width = 140; t5.Height = 30; Controls.Add(t5);

            l1 = new Label(); l1.Text = "ID склада"; l1.Location = new Point(40, 20); l1.Width = 140; l1.Height = 30;
Controls.Add(l1);
            l2 = new Label(); l2.Text = "Название"; l2.Location = new Point(40, 70); l2.Width = 140; l2.Height = 30;
Controls.Add(l2);
            l3 = new Label(); l3.Text = "Адрес"; l3.Location = new Point(40, 120); l3.Width = 140; l3.Height = 30;
Controls.Add(l3);
            l4 = new Label(); l4.Text = "Телефон"; l4.Location = new Point(40, 170); l4.Width = 140; l4.Height = 30;
Controls.Add(l4);
            l5 = new Label(); l5.Text = "Город"; l5.Location = new Point(40, 220); l5.Width = 140; l5.Height = 30;
Controls.Add(l5);
        }

        private void b_click(object sender, EventArgs e)
        {
            con.Open();
            cmd = new OleDbCommand("insert into sklad values(" + t1.Text + ",'" + t2.Text + "','" + t3.Text + "'," +
t4.Text + ",'" + t5.Text + "')", con);
            cmd.ExecuteNonQuery();
            con.Close();
            b_click(null, null);
            {
                MessageBox.Show("Запись добавлена!", "Добавление прошло успешно");
            }
        }
    }
    class Form4 : Form
    {
        Form2 pf;
        Button bt1, bt2, bt3, bt4, bt5;
        TextBox tb1;
        Label l1;
        DataGridView dgv;
        public Form4(Form2 f)
        {
            pf = f; Text = "Заказчики"; Width = 700; Height = 350;

            dgv = new DataGridView(); dgv.Location = new Point(150, 0); dgv.Width = 550; dgv.Height = 250; Con-
trols.Add(dgv);

            bt1 = new Button(); bt1.Text = "Просмотреть таблицу"; bt1.Location = new Point(0, 0); bt1.Width = 140;
            bt1.Height = 30; Controls.Add(bt1); bt1.Click += bt1_click;

            bt2 = new Button(); bt2.Text = "Добавить заказчика"; bt2.Location = new Point(0, 50); bt2.Width = 140;
            bt2.Height = 30; Controls.Add(bt2); bt2.Click += bt2_click;

            l1 = new Label(); l1.Text = "Введите название"; l1.Location = new Point(0, 90); l1.Width = 80; l1.Height
= 20; Controls.Add(l1);
            tb1 = new TextBox(); tb1.Location = new Point(0, 110); tb1.Width = 80; tb1.Height = 30; Con-
trols.Add(tb1);
            bt3 = new Button(); bt3.Text = "Удалить по названию"; bt3.Location = new Point(0, 140); bt3.Width =
140; bt3.Height = 30; Controls.Add(bt3); bt3.Click += bt3_click;
            bt4 = new Button(); bt4.Text = "Поиск по названию"; bt4.Location = new Point(0, 180); bt4.Width = 140;
bt4.Height = 30; Controls.Add(bt4); bt4.Click += bt4_click;
            bt5 = new Button(); bt5.Text = "Закрыть"; bt5.Location = new Point(550, 255); bt5.Width = 80; bt5.Height
= 30; Controls.Add(bt5); bt5.Click += bt5_click;
        }
```

```
private void bt5_click(object sender, EventArgs e)
        {
            Close();
        }

        private void bt4_click(object sender, EventArgs e)
        {
            OleDbConnection con = new OleDbConnection("Provider=MSDAORA;Data Source=orcl;Persist Security
Info=True;Password=whouse;User ID=whouse");
            con.Open();

            OleDbDataAdapter oda = new OleDbDataAdapter("select * from customer where cust_name = '" +
tb1.Text + "'", con);
            DataTable dt = new DataTable();

            dgv.DataSource = dt;
            oda.Fill(dt);

            con.Close();
        }

        private void bt3_click(object sender, EventArgs e)
        {
            OleDbConnection con = new OleDbConnection("Provider=MSDAORA;Data Source=orcl;Persist Security
Info=True;Password=whouse;User ID=whouse");
            con.Open();
            OleDbCommand cmd = new OleDbCommand("delete from customer where cust_name =" + tb1.Text, con);
            cmd.ExecuteNonQuery();
            con.Close();
            bt3_click(null, null);
            {
                MessageBox.Show("Запись удалена!", "Удаление прошло успешно");
            }
            throw new NotImplementedException();
        }

        private void bt2_click(object sender, EventArgs e)
        {
            this.Hide(); new Form11(this).Show(); Visible = false;
        }
        private void bt1_click(object sender, EventArgs e)
        {
            OleDbConnection con = new OleDbConnection("Provider=MSDAORA;Data Source=orcl;Persist Security
Info=True;Password=whouse;User ID=whouse");
            con.Open();

            OleDbDataAdapter oda = new OleDbDataAdapter("select * from customer", con);
            DataTable dt = new DataTable();

            dgv.DataSource = dt;
            oda.Fill(dt);

            con.Close();
        }
        class Form11 : Form
        {
            Form4 pf;
            Button b;
            Label l1, l2, l3, l4, l5;
            TextBox t1, t2, t3, t4, t5;
```

```
  OleDbCommand cmd;
        OleDbConnection con;
        public Form11(Form4 f)
        {
            pf = f; Text = "Форма для добавления"; Width = 220; Height = 310;
            con = new OleDbConnection("Provider=MSDAORA;Data Source=orcl;Persist Security In-
fo=True;Password=whouse;User ID=whouse");

            b = new Button(); b.Text = "Добавить"; b.Location = new Point(40, 220); b.Width = 80;
            b.Height = 30; Controls.Add(b); b.Click += b_click;

            t1 = new TextBox(); t1.Location = new Point(40, 40); t1.Width = 140; t1.Height = 30; Controls.Add(t1);
            t2 = new TextBox(); t2.Location = new Point(40, 90); t2.Width = 140; t2.Height = 30; Controls.Add(t2);
            t3 = new TextBox(); t3.Location = new Point(40, 140); t3.Width = 140; t3.Height = 30; Con-
trols.Add(t3);
            t4 = new TextBox(); t4.Location = new Point(40, 190); t4.Width = 140; t4.Height = 30; Con-
trols.Add(t4);


            l1 = new Label(); l1.Text = "Название"; l1.Location = new Point(40, 20); l1.Width = 140; l1.Height =
30; Controls.Add(l1);
            l2 = new Label(); l2.Text = "Адрес"; l2.Location = new Point(40, 70); l2.Width = 140; l2.Height = 30;
Controls.Add(l2);
            l3 = new Label(); l3.Text = "Город"; l3.Location = new Point(40, 120); l3.Width = 140; l3.Height = 30;
Controls.Add(l3);
            l4 = new Label(); l4.Text = "Телефон"; l4.Location = new Point(40, 170); l4.Width = 140; l4.Height =
30; Controls.Add(l4);

        }

        private void b_click(object sender, EventArgs e)
        {
            con.Open();
            cmd = new OleDbCommand("insert into customer values('" + t1.Text + "','" + t2.Text + "','" + t3.Text +
"','" + t4.Text + "')", con);
            cmd.ExecuteNonQuery();
            con.Close();
            b_click(null, null);
            {
                MessageBox.Show("Запись добавлена!", "Добавление прошло успешно");
            }
        }
    }
}
```