**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ**

ФГБОУ ВО

**«БРЯНСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**Кафедра** «Информатика и программное обеспечение»

**КУРСОВОЙ ПРОЕКТ**

**по дисциплине «Базы данных»**

**Тема:** **«Система управления хранением данных машиностроительного предприятия»**

Студент гр. О-21-ИВТ1-ПО-Б

Зач. кн.№ 20.0159

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Шпаков В.В.

«11» января 2024 г.

Преподаватель

\_\_\_\_\_\_\_\_\_\_\_\_\_ Копелиович Д.И.

«11» января 2024 г.

**БРЯНСК 2024**

Содержание

[ВВЕДЕНИЕ 3](#_Toc155900343)

[1. Описание предметной области 5](#_Toc155900344)

[1.1. Функциональные требования 5](#_Toc155900345)

[1.2. Инструментальные средства разработки 6](#_Toc155900346)

[2. Конструкторская часть 7](#_Toc155900347)

[2.1. Концептуальная модель базы данных 7](#_Toc155900348)

[2.2. Создание таблиц в базе данных 8](#_Toc155900349)

[2.3. Восстановление базы данных 12](#_Toc155900350)

[2.4. Администрирование базы данных 14](#_Toc155900351)

[2.5. SQL-запросы 15](#_Toc155900352)

[2.6. Хранимые процедуры 17](#_Toc155900353)

[2.7. Триггеры 21](#_Toc155900354)

[3. Экспериментальная часть 22](#_Toc155900355)

[Заключение 24](#_Toc155900356)

[Список литературы 25](#_Toc155900357)

# ВВЕДЕНИЕ

Целью данной курсовой работы является создание базы данных для эффективного учета данных на машиностроительном предприятии. Актуальность исследования обусловлена возможностью удобного хранения и управления информацией о продукции, материалах, заказах, а также связях между ними. База данных предоставляет инструмент для создания, изменения и удаления записей, что обеспечивает эффективное ведение бизнес-процессов на предприятии.

База данных представляет собой организованный набор данных, специфических для машиностроительной отрасли, предназначенный для долгосрочного хранения во внешней памяти компьютера. Система управления базами данных (СУБД) включает ядро, процессор языка БД, подсистемы поддержки времени исполнения и вспомогательное ПО. СУБД обеспечивает управление данными, взаимодействие с информацией, логирование и поддержку различных форматов баз данных.

Основные задачи работы включают:

* Анализ предметной области и формулирование требований к базе данных для машиностроительного предприятия.
* Проектирование инфологической модели базы данных, отражающей структуру и взаимосвязи между данными.
* Создание базы данных с использованием СУБД PostgreSQL и разработка графического интерфейса для удобного взаимодействия пользователя с данными.
* Предусмотрение возможности восстановления базы данных в случае сбоев или потери данных.
* Тестирование и отладка программного продукта для обеспечения его корректной работы.

Объектом исследования является процесс разработки базы данных для учета материалов, продукции и брака на машиностроительном предприятии. Предметом исследования является использование средств СУБД **PostgreSQL** и интегрированной среды разработки **Visual Studio 2022**.

Работа над курсовой предполагает создание эффективной и удобной системы учета данных, которая соответствует особенностям машиностроительной отрасли и облегчает бизнес-процессы на предприятии.

# Описание предметной области

Организация производства на машиностроительном предприятии включает в себя несколько ключевых сущностей, каждая из которых играет важную роль в бизнес-процессах. Среди них: материалы, продукция, брак, клиенты, заказы и партии. Процесс производства начинается с выбора и закупки материалов, формирования сотрудников, учета сложности, оформления заказов и их выполнения.

## Функциональные требования

Для данной курсовой работы необходимо реализовать:

* создание концептуальной модели базы данных;
* создание непосредственно базы данных в СУБД, организация таблиц с конкретными типами данных и с ограничениями;
* организация бекапов;
* организация работы с пользователями и ролями;
* использование sql-запросов для взаимодействия с таблицами базы данных;
* использование хранимых процедур для переиспользуемого взаимодействия;
* использование триггеров для наблюдения за таблицами, и непосредственного редактирования, при изменении какой-либо таблицы из базы данных;
* создание простейшего интерфейса для созданной базы данных, который включает в себя: создание, редактирование, удаление, просмотр для каждой таблицы и непосредственное переключение между таблицами.

## Инструментальные средства разработки

Программная реализация представляет собой настольное приложение, разработанное на основе технологии создания приложений Microsoft .NET с использованием языка программирования C#. В качестве СУБД для реализации хранилища данных, был использован PostgreSQ, pgAdmin4.

Microsoft .NET – программная технология, для создания как обычных программ, так и веб-приложений, в качестве платформы для разработок создана фирмой Microsoft. Одной из основных идей Microsoft .NET является совместимость различных приложений, написанных на разных языках.

Windows Forms в C# - это технология разработки графического интерфейса (GUI) для приложений под операционную систему Windows. Она позволяет создавать интерактивные и пользовательские приложения с использованием различных элементов управления, таких как кнопки, текстовые поля, таблицы и другие.

Npgsql - это библиотека для работы с базами данных PostgreSQL в языке программирования C#. Npgsql предоставляет удобные классы и методы для подключения к серверу PostgreSQL, выполнения запросов, получения и обработки данных.

# Конструкторская часть

## Концептуальная модель базы данных

Концептуальная модель базы данных представляет собой наглядную диаграмму, использующую общепринятые обозначения и подробно демонстрирующую взаимосвязи между объектами и их характеристиками. Она создается с целью последующего проектирования базы данных, включая возможность преобразования её в реляционную форму. На концептуальной модели визуально отображаются связи между объектами данных и их атрибутами.

Для обеспечения единообразия программирования баз данных введены следующие понятия концептуальных баз данных:

**Объект или сущность:** Фактический или абстрактный объект, имеющий значение для конкретной предметной области. Пример - физический объект или человек, наблюдение за которым представляет интерес для заказчика.

**Атрибут:** Характеристика объекта, соответствующая его сущности. Это свойство объекта, используемое для его идентификации, упорядочивания или описания состояния.

**Связь (отношение):** Взаимосвязь между объектами, показывающая, как атрибуты одной сущности связаны с атрибутами другой. Связь описывает взаимодействие между объектами.

Для проектирования инфологической модели часто используется утилита Draw.io - программное обеспечение для моделирования данных, которое обеспечивает удобные средства построения диаграмм для анализа потоков данных и их управления в любой системе.

Сущность в ER-модели обязана иметь уникальный идентификатор, отличающийся от других объектов того же типа. Этот идентификатор обладает уникальным именем и свойствами, а сущность сама обладает атрибутами, которые либо принадлежат ей, либо приобретаются через связи с другими сущностями. Связи между сущностями могут быть одним-к-одному, одним-ко-многим, многими-ко-многими, идентифицирующими или неидентифицирующими, в зависимости от их характера и взаимодействия.

Концептуальная модель базы данных для цеха представлена на рис. 1.

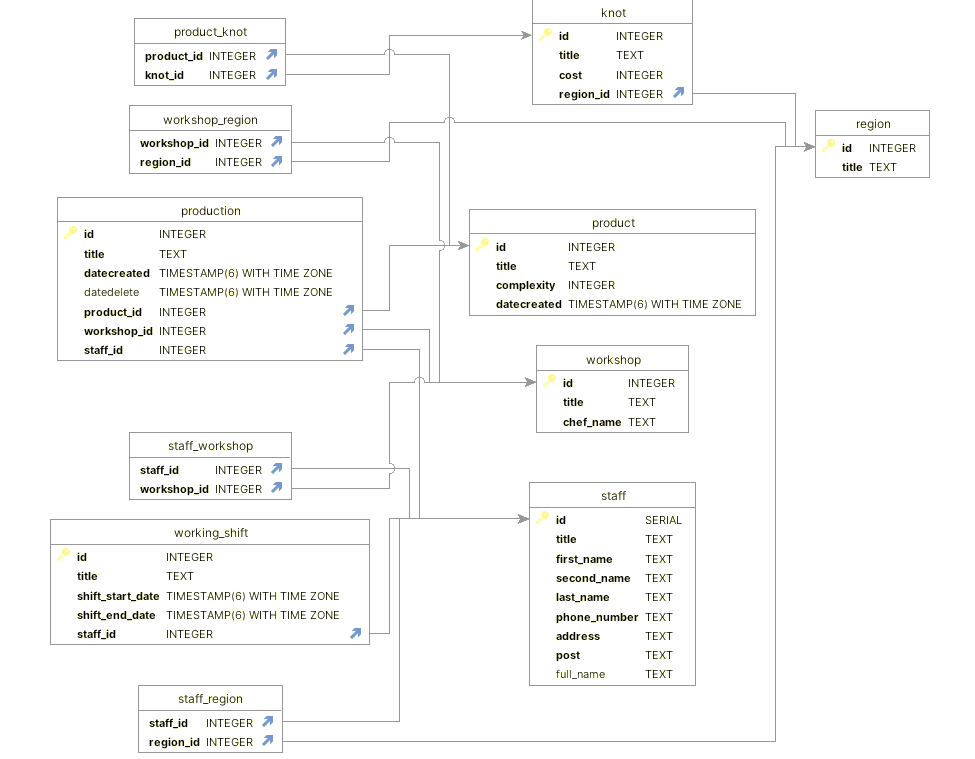


Рис. . Концептуальная модель базы данных

## Создание таблиц в базе данных

Чтобы создать таблицы можно использовать инструменты СУБД pgAdmin4 или использовать sql-запросы CREATE TABLE.

Создание всех таблиц в данной курсовой работе организовано один sql-запросом (см. листинг 1).

Ограничение таблиц организовано с помощью отдельных ключевых слов:

* PRIMARY KEY – ограничение устанавливает первичный ключ на поле идентификатора;
* NOT NULL – ограничение устанавливается на поля, которые обязательны к заполнению;
* CHECK – ограничение на ввод и длину данных (поля, где вводятся только цифры или только буквы, а также поля с фиксированной длиной);
* UNIQUE – ограничение устанавливающиеся на поля, которые могут иметь только уникальное значение и не могут повторяться, например такие, как телефон и email;
* CONSTRAINT <имя ограничения> - задает имя ограничения, которое реализует каскадное удаление.

**Листинг 1**

**Запрос с созданием и ограничением таблиц**

|  |
| --- |
| create schema production;  create table production.region  (  id integer primary key generated by default as identity,  title text not null  );  create table production.product  (  id integer primary key generated by default as identity,  title text not null,  complexity int not null,  DateCreated timestamp(6) with time zone not null  );  create table production.knot  (  id integer primary key generated by default as identity,  title text not null,  cost integer NOT NULL CHECK (cost >= 0),  region\_id int not null,  foreign key(region\_id) references production.region(id)  on delete cascade  );  create table production.workshop  (  id integer primary key generated by default as identity,  title text not null,  chef\_name text not null  );  create table production.staff  (  id serial PRIMARY KEY,  title text NOT NULL,  first\_name text NOT NULL,  second\_name text NOT NULL,  last\_name text NOT NULL,  phone\_number text NOT NULL UNIQUE,  address text NOT NULL UNIQUE,  post text NOT NULL UNIQUE,  full\_name text GENERATED ALWAYS AS (  first\_name || ' ' || LEFT(second\_name, 1) || '.' || LEFT(last\_name, 1) || '.'  ) STORED  ); |

|  |
| --- |
| create table production.production  (  id integer primary key generated by default as identity,  title text not null,  DateCreated timestamp(6) with time zone not null,  DateDelete timestamp(6) with time zone,  product\_id int not null,  workshop\_id int not null,  staff\_id int not null,  foreign key(product\_id) references production.product(id)  on delete cascade,  foreign key(workshop\_id) references production.workshop(id)  on delete cascade,  foreign key(staff\_id) references production.staff(id)  on delete cascade  );  create table production.product\_knot  (  product\_id int not null,  knot\_id int not null,  foreign key(product\_id) references production.product(id)  on delete cascade,  foreign key(knot\_id) references production.knot(id)  on delete cascade  );  create table production.workshop\_region  (  workshop\_id int not null,  region\_id int not null,  foreign key(workshop\_id) references production.workshop(id)  on delete cascade,  foreign key(region\_id) references production.region(id)  on delete cascade  );  create table production.working\_shift  (  id integer primary key generated by default as identity,  title text not null,  shift\_start\_date timestamp(6) with time zone not null,  shift\_end\_date timestamp(6) with time zone not null,  staff\_id int not null,  foreign key(staff\_id) references production.staff(id)  on delete cascade  );  create table production.staff\_region  (  staff\_id int not null,  region\_id int not null,  foreign key(staff\_id) references production.staff(id)  on delete cascade,  foreign key(region\_id) references production.region(id)  on delete cascade  );  create table production.staff\_workshop  (  staff\_id int not null,  workshop\_id int not null,  foreign key(staff\_id) references production.staff(id)  on delete cascade,  foreign key(workshop\_id) references production.workshop(id)  on delete cascade  ); |

## Восстановление базы данных

Резервному копированию баз данных в PosgreSQL отводится огромное значение. Правильно настроенное, оно поможет уберечь базу данных от повреждений и потери данных.

Для создания резервной копии, требуется зайти в СУБД pgAdmin4, выбрать нужную базу данных, нажать правую кнопку мыши, выбрать пункт, Резервная копия.

На рис. 2 представлено создание резервной копии.

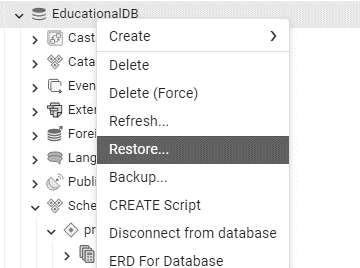


Рис. . Резервное копирование базы данных

Для того, чтобы восстановить базу данных необходимо выбрать Восстановить (см. рис. 3) и выбрать необходимые параметры восстановления.

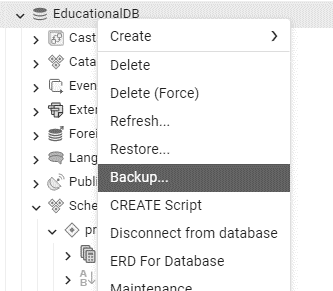


Рис. . Восстановление базы данных

Для проверки работоспособности резервного копирования и восстановления базы данных, создадим бекап с базой данных, изменив какую-то таблицу и восстановим базы данных.

Исходные данные представлены на рис. 4.

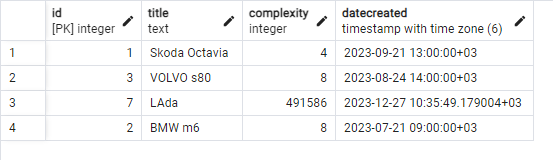


Рис. 4. Исходные данные таблицы product

Создадим бекап данного состояния, а потом изменим какое-нибудь поле. Бекап представлен на рис. 5.

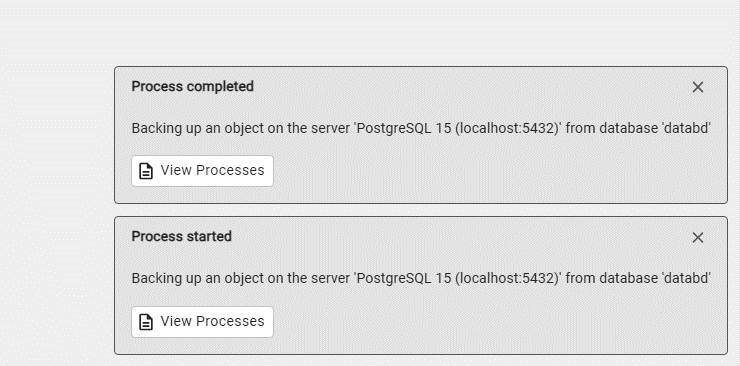


Рис. 5. Создание полного бекапа

Изменённая таблица представлена на рис. 6.

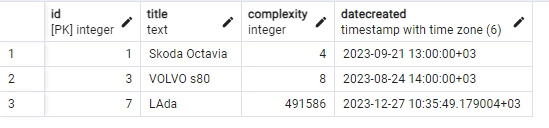


Рис. 6. Таблица с изменёнными данными product

После восстановления полного бекапа необходимо обновить все таблицы, чтобы изменения вступили в силу. Восстановленная таблица представлена на рис. 7.

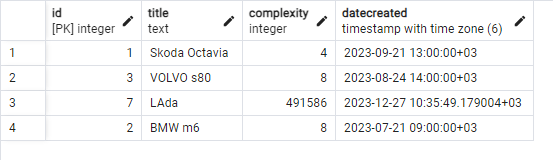


Рис. 7. Восстановленная таблица product

Видим, что таблица корректно восстановилась, поэтому бекапы работают так, как и ожидалось.

## Администрирование базы данных

Для того, чтобы создать аккаунт, необходимо сначала создать имя для входа (см. рис. 8). В последствии ему необходимо назначит роли, соотнесение к каким-то таблицам, защищаемые ресурсы и его состояние. Этим может заниматься только администратор базы данных. Следующим этапом необходимо создать пользователя в нашей базе данных MyDataBase. Для этого администратор должен зайти в безопасность/пользователи и создать пользователя, у которого могут быть свои настройки с безопасностью и доступом.

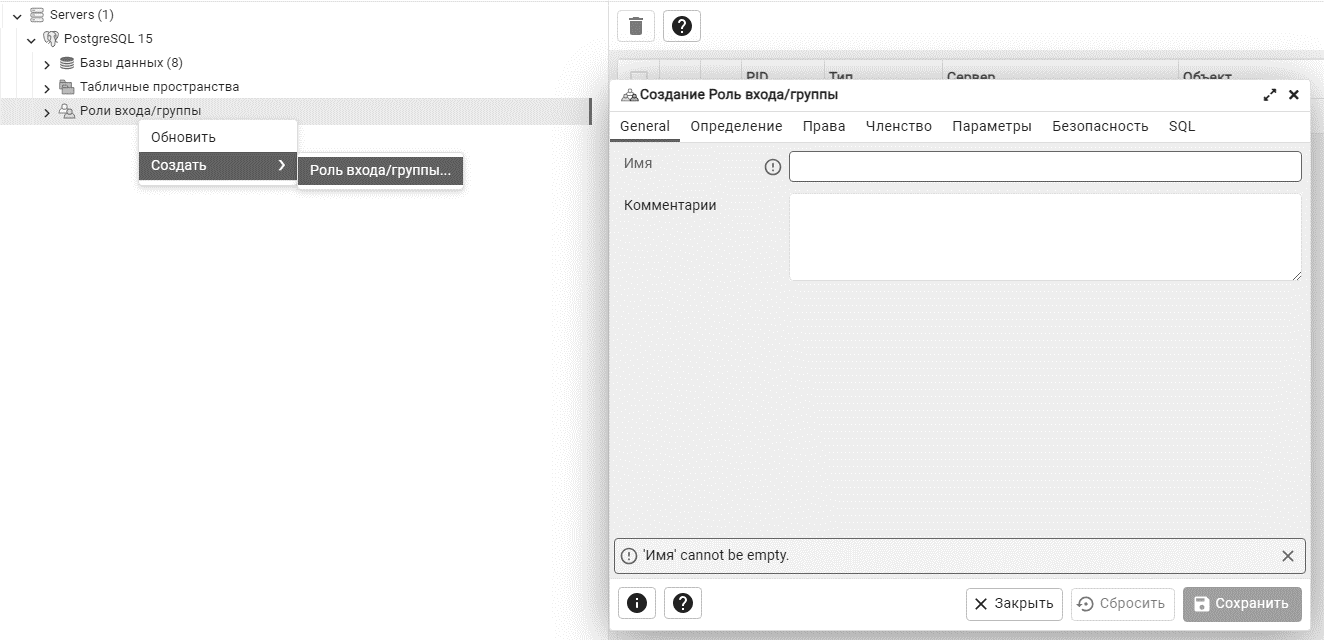


Рис. 8. Создание имени для входа

Чтобы зайти под созданным пользователем необходимо отсоединить базы данных и включить проверку подлинности pgAdmin4.

## SQL-запросы

В данной курсовой работе необходимо реализовать следующие запросы:

* запрос с выборкой полей и условием;
* запрос с использованием DISTINCT;
* запрос с использованием специальных операторов SQL;
* запрос с единичной агрегатной функцией;
* запрос с агрегатной функцией, группировкой и использованием условия;
* запрос с внутренним соединением двух таблиц;
* запрос, использующий одно из внешних соединений двух таблиц;
* составной запрос;
* составные запросы с использованием операторов ANY, ALL, EXISTS;
* соотнесённый запрос;
* запрос ввода данных, удаление, изменение.

Запросы с выборкой и условием, с применением DISTINCT, LIKE представлены на рис. 9

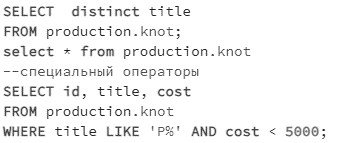


Рис.9. Запросы с выборкой

Запрос с агрегатной функцией представлен на рис. 10.



Рис. 10. Запрос с агрегатной функцией

Запрос с группировкой представлен на рис. 11.

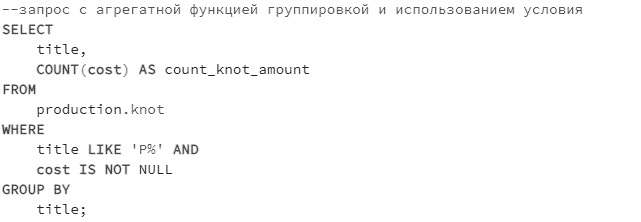


Рис. 11. Запросы с группировкой

Запрос с ключевыми словами ANY представлены на рис. 12.

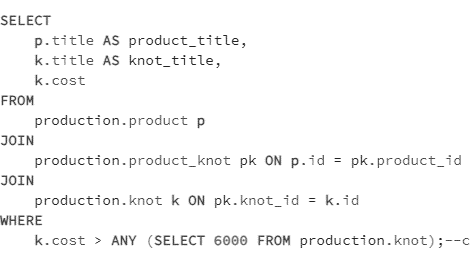


Рис. 12. Запрос с ключевыми словами ANY

Запросы с добавлением, обновлением и удалением представлены на  
рис. 13.

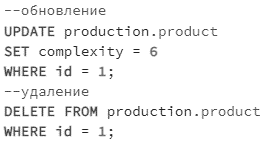


Рис. . Запросы с добавлением, обновлением, удалением

## Хранимые процедуры

Часто операции с данными требуют выполнения нескольких инструкций в определенной последовательности. Например, при добавлении информации о покупке товара необходимо сначала проверить наличие товара, возможно, удовлетворяя еще несколько дополнительных условий. В таких случаях эффективным решением является использование хранимых процедур.

Хранимые процедуры представляют из себя совокупность инструкций, выполняемых как единое целое. Их применение позволяет упростить сложные операции, объединив их в один объект. Например, процесс покупки товара может быть абстрагирован в виде хранимой процедуры, что упрощает изменения и обслуживание кода.

Важным преимуществом хранимых процедур является возможность ограничения доступа к данным в таблицах, что снижает риск нежелательных действий с данными. Такой подход способствует управлению безопасностью данных.

Помимо этого, хранимые процедуры обеспечивают повышенную производительность. Они обычно выполняются быстрее, чем обычные SQL-инструкции, поскольку код процедур компилируется один раз при первом запуске и сохраняется в скомпилированной форме.

Создание хранимой процедуры осуществляется с использованием команды CREATE PROCEDURE или CREATE PROC.

Создание процедуры для добавления продукта на листинге листинг 2.

**Листинг 2**

**Создание хранимой процедуры для добавления нового продукта**

|  |
| --- |
| create or replace function production.add\_new\_product(  p\_title text,  p\_complexity int,  p\_knots integer[]  )returns integer as $$  declare  v\_product\_id integer;  v\_total\_cost integer := 0;  begin  -- Добавление нового продукта  insert into production.product(title, complexity, DateCreated)  values (p\_title, p\_complexity, NOW())  returning id into v\_product\_id;  -- Связывание узлов с продуктом в таблице product\_knot  for i in 1..array\_length(p\_knots, 1) loop  insert into production.product\_knot(product\_id, knot\_id)  values (v\_product\_id, p\_knots[i]);  end loop;  -- Рассчет общей стоимости узлов для нового продукта  select into v\_total\_cost  sum(k.cost)  from production.product\_knot pk  join production.knot k ON pk.knot\_id = k.id  where pk.product\_id = v\_product\_id;  -- Обновление общей стоимости в таблице product  update production.product  set complexity = v\_total\_cost  where id = v\_product\_id;  -- Возвращение общей стоимости нового продукта  return v\_total\_cost;  end;  $$ language plpgsql; |

Создание хранимой процедуры для добавления нового сотрудника листинг 3.

**Листинг 3**

**Создание хранимой для добавления нового сотрудника**

|  |
| --- |
| create or replace function production.add\_new\_staff\_with\_shift(  p\_title text,  p\_first\_name text,  p\_second\_name text,  p\_last\_name text,  p\_phone\_number text,  p\_address text,  p\_post text,  p\_shift\_title text,  p\_shift\_start\_date timestamp with time zone,  p\_shift\_end\_date timestamp with time zone  ) returns text as $$  declare  v\_staff\_id integer;  v\_full\_name text;  begin  -- Добавление нового сотрудника  insert into production.staff(title, first\_name, second\_name, last\_name, phone\_number, address, post)  values (p\_title, p\_first\_name, p\_second\_name, p\_last\_name, p\_phone\_number, p\_address, p\_post)  returning id into v\_staff\_id;  -- Добавление смены для нового сотрудника  insert into production.working\_shift(title, shift\_start\_date, shift\_end\_date, staff\_id)  values (p\_shift\_title, p\_shift\_start\_date, p\_shift\_end\_date, v\_staff\_id);  -- Получение полного имени сотрудника  select info v\_full\_name  first\_name || ' ' || coalesce(second\_name || ' ', '') || last\_name  from production.staff  where id = v\_staff\_id;  -- Возвращение полного имени нового сотрудника  return v\_full\_name;  end;  $$ language plpgsql; |

Чтобы вызвать хранимую процедуру необходимо использовать ключевое слово CALL.

Вызов хранимой процедуры update\_product представлен на листинге 4.

**Листинг 4**

**Вызов хранимой процедуры production.add\_new\_product**

|  |
| --- |
| select production.add\_new\_product('Lada Vesta', 3, ARRAY[1, 2, 3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18]); |

Вызов хранимой процедуры add\_material представлен на листинге 5.

**Листинг 5**

**Вызов хранимой процедуры production.add\_new\_staff\_with\_shift**

|  |
| --- |
| select production.add\_new\_staff\_with\_shift('Умный парень', 'Кирил', 'И.', 'Финов', '12345682149', 'Ленинград', 'Поэт', 'Day Shift', NOW(), NOW() + interval '8 hours'); |

## Триггеры

Триггеры представляют специальный тип хранимой процедуры, которая вызывается автоматически при выполнении определенного действия над таблицей или представлением, в частности, при добавлении, изменении или удалении данных, то есть при выполнении команд INSERT, UPDATE, DELETE.

Для создания триггера применяется выражение CREATE TRIGGER, после которого идет имя триггера. Как правило, имя триггера отражает тип операций и имя таблицы, над которой производится операция.

Каждый триггер ассоциируется с определенной таблицей или представлением, имя которых указывается после слова ON.

Затем устанавливается тип триггера. Мы можем использовать один из двух типов:

* AFTER: выполняется после выполнения действия. Определяется только для таблиц.
* INSTEAD OF: выполняется вместо действия. Определяется для таблиц и представлений

После типа триггера идет указание операции, для которой определяется триггер: INSERT, UPDATE или DELETE.

Триггер на создание связей меж таблицами продукты и материалы (см. рис. 14).

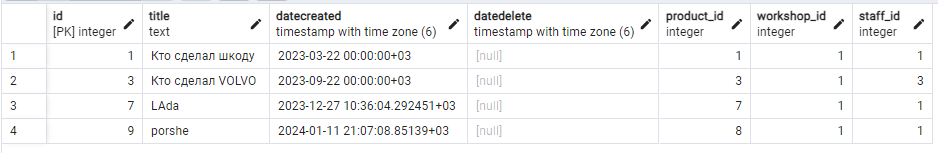


Рис. 14. Работа триггера при создании нового продукта

# Экспериментальная часть

В качестве экспериментальной части база данных была обёрнуто Windows Form на языке программирования C#.

Пользователь может перемещаться по таблицам базы по верхнему меню, а также добавлять записи при нажатии на кнопку Добавить.

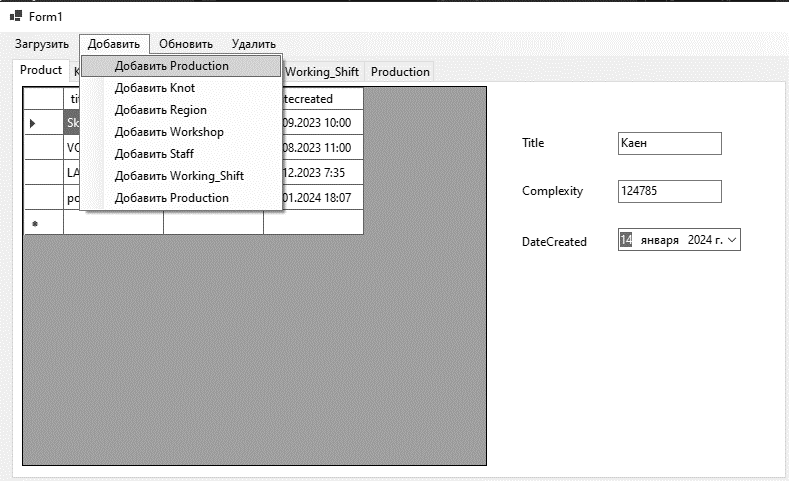


Рис. 5. Добавление данных в таблицу

Удаление записи.

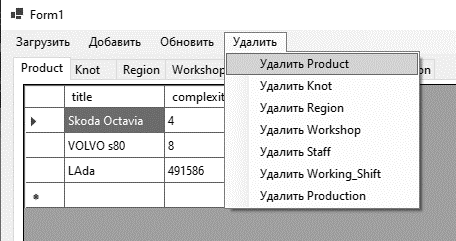


Рис. . Удаление записи в таблице

Изменение данных в таблице.

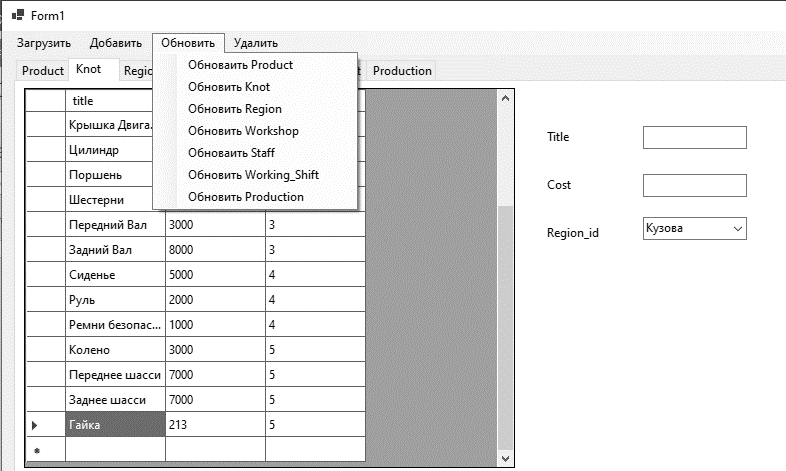


Рис. 17. Изменение записи в таблице

# Заключение

В рамках курсовой работы была разработана база данных для организации учета материалов, товара, брака и заказов для производства мебели, тем самым реализовав поставленную цель. В ходе исследования были выполнены следующие задачи:

В рамках выполнения курсовой работы была успешно разработана база данных для эффективного учета материалов, товаров, брака и заказов, предназначенных для производства мебели. Реализация поставленной цели включала выполнение следующих задач:

1. Осуществлен анализ предметной области с последующим формированием требований к базе данных.
2. Представлена характеристика используемых инструментальных средств при разработке программного продукта.
3. Создана полноценная база данных вместе с пользовательским интерфейсом, обеспечивающим удобное взаимодействие с ней.
4. Проведены тщательное тестирование и отладка программного продукта, что подтверждает его функциональность и отсутствие ошибок.

Представленная версия базы данных является завершенным программным продуктом, успешно выполняющим все необходимые функции без сбоев и ошибок.

# Список литературы

1. ГОСТ Р ИСО/МЭК 12207- 2010 «Системная и программная инженерия. Процессы жизненного цикла программных средств».
2. Агальцов В.П. Базы данных. В 2-х кн. Книга 2. Распределенные и удаленные базы данных [Электронный ресурс]: Учебник / В.П. Агальцов. - М.: ФОРУМ: ИНФРА-М, 2018. - 271 с.
3. Базы данных: Учебник / Шустова Л.И., Тараканов О.В. - М.: НИЦ ИНФРА-М, 2019. - 304 с.
4. Белладжио Миллиган. Разработка программного обеспечения: управление изменениями. – М.: [ДМК-Пресс](https://www.labirint.ru/pubhouse/1416/), 2019. – 384 с.
5. Гринченко, Н.Н. Проектирование баз данных. СУБД Microsoft Access: Учебное пособие для вузов. / Н.Н. Гринченко и др. - М.: РиС, 2013. - 240 c.
6. Коннолли, Т. Базы данных. Проектирование, реализация и сопровождение. Теория и практика / Т. Коннолли. - М.: Вильямс И.Д., 2017. - 1440 c.
7. Лукин, В.Н. Введение в проектирование баз данных / В.Н. Лукин. - М.: Вузовская книга, 2015. - 144 c.
8. "PostgreSQL. Администрирование и разработка" - авторы Алексей Лесовой, Олег Бартунов, Илья Куликов. 2019. - 174 c.